AD-756 970

ARTIFICIAL INTELLIGENCE -- RESEARCH AND APPLICATIONS

Peter E. Hart, et al

Stanford Research Institute

Prepared for:

Army Research Office-Durham
Advanced Research Projects Agency

December 1972

Annual Technical Report

ARTIFICIAL INTELLIGENCE—RESEARCH AND APPLICATIONS

By: P. E. HART R. E. FIKES T. D. GARVEY N. J. NILSSON
D. NITZAN J. M. TENENBAUM B. M. WILBER

Prepared for:

ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

CONTRACT DAHC04-72-C-0008

Approved for public release; distribution unlimited.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Stanford Research Institute 333 Ravenswood Avenue Menlo Park, California 94025	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP n/a

3 REPORT TITLE

ARTIFICIAL INTELLIGENCE--RESEARCH AND APPLICATIONS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Annual Technical Report: 8 October 1971 through 8 October 1972.

5 AUTHOR(S) (First name, middle initial, last name)

Peter E. Hart, Richard E. Fikes, Thomas D. Garvey, Nils J. Nilsson, David Nitzan, J. Martin Tenenbaum, and B. Michael Wilber.

6 REPORT DATE December 1972	7a. TOTAL NO. OF PAGES<br~~136~~ 128	7b. NO. OF REFS 10
8a. CONTRACT OR GRANT NO. DAHC04-72-C-0008 b. PROJECT NO. c. Program Code Nc. 2D30 d. ARPA Order No. 1943	9a. ORIGINATOR'S REPORT NUMBER(S) SRI Project 1530 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AROD - 10452.4-A	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency Arlington, Virginia 22209

13. ABSTRACT

This report describes activities during the most recent year of a program of research in artificial intelligence. During the year a number of experiments were conducted with an existing system for the control of a robot that autonomously plans, learns, and carries out tasks in a real laboratory environment. Concomitantly, designs for a new robot system were evolving. Of particular interest is a conceptual design for a novel perceptual subsystem, and some preliminary thoughts on the design of hierarchical problem solvers.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer-controlled robot						
Automatic problem solving						
Automatic perception						
Multisensory perception						

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

Form Approved
Budget Bureau No. 22-R0293
December 1972

Annual Technical Report
Covering the Period 8 October 1971 through 8 October 1972
Stanford Research Institute Project 1530

ARTIFICIAL INTELLIGENCE—RESEARCH AND APPLICATIONS

Submitted by

PETER E. HART
Project Leader
(415) 326-6200, Ext. 2129

Authors

P. E. HART R. E. FIKES T. D. GARVEY N. J. NILSSON
D. NITZAN J. M. TENENBAUM B. M. WILBER

CONTRACT DAHC04-72-C-0008
ARPA Order Number 1943
Program Code Number 2D30

Effective Date of Contract: 8 October 1971
Contract Expiration Date: 9 October 1973
Amount of Contract: $1,191,607.00

Prepared for

ADVANCED RESEARCH PROJECTS AGENCY
ARLINGTON, VIRGINIA 22209

Approved by:

BERTRAM RAPHAEL, *Director*
Artificial Intelligence Center

BONNAR COX, *Executive Director*
Information Science and Engineering Division

Copy No. 15

CONTENTS

ABSTRACT iii

LIST OF ILLUSTRATIONS. ix

LIST OF TABLES xi

GLOSSARY xiii

I INTRODUCTION. 1

A. General. 1

B. Background 1

C. Report Outline 2

II PROBLEM SOLVING 3

A. Introduction 3

B. Description of Work on the STRIPS-PLANEX System. . . . 4

1. Development and Refinements 4
2. Experimental Results. 6

C. Preliminary Specifications for a New Problem Solving System 16

1. Criteria for the New System 16
2. Hierarchical Planning 19

D. Multirobot Experiments 22

III PERCEPTION. 25

A. Introduction 25

1. Perceptual Philosophy 25
2. Design Considerations 26
3. Organization of this Section. 29

B. Overview of System 29

1. Scenario. 29
2. Perceptual Strategy 34

III PERCEPTION (continued)

C. Conceptual Design. 38

1. Introduction. 38
2. Definitions 40
3. Planning. 42
4. Execution 52

D. Goal Directed Scene Segmentation 65

1. Background. 66
2. Classification Approach 67
3. Coping with Texture 69
4. Operational Details 70
5. Error Recovery. 71

E. Multisensory Data. 74

1. Analysis of Range Data. 75
2. Effects of Errors in Range Data 86
3. Line Fitting of Region Boundary Points. 86
4. Analysis of Color Data. 87
5. Generation of Test Data 88

F. Research Methodology 89

1. System Features 90
2. Plans 91

IV SYSTEM SOFTWARE 95

A. Introduction 95
B. Utility Factors. 95

1. Comprehensibility 95
2. Speed 96
3. Reliability 97

C. Translation to BBN LISP. 97

1. Overview. 97
2. Forking 98
3. Additions for TRANSOR 99

V SYSTEM HARDWARE 101

A. Introduction 101

B. A Time-of-Flight Range Scanner 101

1. Introduction. 101

V SYSTEM HARDWARE (continued)

2. The Experimental Model of the Range Scanner . . . 102
3. Calculation of Sensitivity. 106

C. A Triangulation Range Finder 108

D. A Radar Motion Detector. 110

E. Unimate Arm. 110

F. Current System Configuration 111

VI PUBLICATIONS AND PRESENTATIONS. 115

A. Publications 115

B. Presentations. 116

REFERENCES 119

DD Form 1473

ILLUSTRATIONS

1	Map of Shakey's Experimental Environment.	7
2	Example Problem 1	10
3	Example Problem 2	11
4	Example Problem 3	12
5	Example Problem 4	13
6	Example Problem 5	14
7	A Simplified Office Environment Representing our Experimental Domain	31
8	Visual Memory Hierarchy	42
9	Planning Tree for Finding OBJECT1	43
10	Extended Planning Tree for OBJECT1.	45
11	Planning Tree for Finding Telephone	59
12	Telephone Planning Tree After Failure of Dark Gray Color Detector.	61
13	Tree After Finding Floor.	62
14	Tree Resulting from Location of Horizontal Plane.	64
15	Tree After Finding Table.	65
16	Range-Finder Centered Image Coordinates	77
17	Computed Floor Boundary	80
18	Range Values with Constant-φ_p Scan of a Horizontal Surface Below the Horizon	81
19	Surface and Intersection Boundaries	82
20	End-Point $r(\beta_p)$ Fitting Method.	84
21	Multiple $r(\beta_p)$ Planar-Region Segments of a Concave and Convex Surface.	85
22	Scanning Range Finder--Simplified	103
23	Scanning Range Finder--Detailed	104

24 Analog Output Function. 107

25 A Triangulation Optical Range Finder. 109

26 PDP-15/Unimate Communications: Block Diagram 111

27 PDP-15/Unimate Communications: Unimate Control Logic . . . 112

28 SRI Artificial Intelligence Center Computer System. 113

TABLES

1 Operator Descriptions 9

2 Statistics for STRIPS Behavior. 15

3 Initial World Model 32

4 Attributes and Contextural Relations for Acquisition. . . . 36

GLOSSARY

DFOM	direct figure of merit
FOM	figure of merit
GM	goal monitor
I	interest
IFOM	indirect figure of merit
MACROP	macro operator
NI	normalized interest
PLANEX	name of a plan executing system
SHAKEY	name of SRI's mobile robot
SRI	Stanford Research Institute
STRIPS	name of a plan generating system
STWM	short term world model
VM	visual memory

I INTRODUCTION

A. General

This annual report describes the work performed during the most recent year of a program of research in the field of artificial intelligence. The work reported here began in October 1971, and is itself the continuation of work performed under a previous contract.* Therefore this is a report on the most recent accomplishments and status of a continuing research program.

During the course of the year we have documented the details of our technical work in a series of reports, journal articles, and presentations.† Our intention in this report is to provide an overview of the project, rather than to reproduce those details here.

B. Background

For a number of years our work has been focused on the application of artificial intelligence techniques to the control of a mobile automaton--a "robot" nicknamed Shakey--in an actual laboratory environment. This work reached its first plateau in 1969 with the completion of the first integrated robot system: a mobile vehicle equipped with a TV camera and other sensors, and controlled by an SDS 940 computer. During the following two and a half years we developed a new, more powerful robot system. While the robot vehicle remained substantially unchanged, the old SDS 940 computer was replaced by a PDP-10/PDP-15 facility with

---

*Contract NASW-2164.

†These are listed in Section VI.

significantly more capability. The software controlling the robot was completely redesigned to incorporate more general and powerful methods both for solving robot problems in the abstract and also for executing the solutions in the real world. This second phase of our activity reached a plateau during the past year, by which time we had completed the design and implementation of the entire system and had carried out a series of experiments to explore its strengths and weaknesses.

As this work progressed we developed preliminary ideas for methods that would dramatically increase the capabilities of the robot in several directions. It was clear that these methods could not be included in the existing system in any convenient way. Accordingly, we elected to devote most of our resources to the task of designing a new robot system that would encompass our new ideas for robot problem solving, perception, and real-world execution monitoring. Thus, our work during the past year has been divided between completing some tasks associated with the existing robot software and beginning design studies for a new system.

C. Report Outline

Section II of this report presents our recent work in robot problem solving. It describes some experiments performed on the existing system, and outlines some of our ideas for a new problem solver. Section III discusses the design of a robot perception system for analyzing pictorial and range data. In Sections IV and V we describe the software and hardware support activities associated with our research. Finally, Section VI lists the publications and presentations that were prepared or presented during the project period.

II PROBLEM SOLVING

A. Introduction

Our research on automatic problem solving has as its goal the development of systems that can plan and execute sequences of actions for a robot. In our formulation, we assume that the robot is given some command such as "push the small box next to the large box," and the problem solver then creates a plan for accomplishing the task. The plan consists of a sequence of motor-action programs, such as "move to position X = 3.2, Y = 4.6." After a plan is generated, we desire that it be executed "intelligently," that is, with due regard for the actual effects of each action.

During the last two years we have developed a plan generating system called STRIPS, a plan executing system called PLANEX, and a learning system that generalizes and saves plans produced by STRIPS. The present status of these systems is well documented in Ref. (1),* so we shall not describe details here. We have also produced a 25-minute, 16-mm, sound film[2] that depicts STRIPS and PLANEX in action controlling our mobile robot, SHAKEY. During the past year we have made some improvements to these systems and conducted some experiments illustrating performance on some learning tasks. These developments are discussed in more detail in the next section.

Our work with STRIPS has clarified some of its limitations, and we have begun to think about how they might be overcome. The results of some of our speculations on this subject are contained in a paper[3] given

---

*References are listed at the end of this report.

at the last Machine Intelligence workshop. Recently we have begun the design of a new problem solving system. Progress on this design will be described later in this report.

B. Description of Work on the STRIPS-PLANEX System

1. Development and Refinements

During the preceding year we continued development and experimentation with plan generalization procedures for the STRIPS-PLANEX system. We can illustrate some of the issues we have dealt with by considering the following example. Assume that adjacent rooms R1 and R2 are connected by door D1, the robot is in room R1, box B1 is in room R2, and the task is to bring box B1 into room R1.

If STRIPS had available the appropriate GOTHRU and PUSHTHRU operators, then it could form the two step plan:

GOTHRU(D1,R1,R2)	[Go through door D1 from room R1 into room R2]
PUSHTHRU(B1,D1,R2,R1)	[Push B1 through door D1 from R2 into room R1].

While this sequence solves the original task, it probably doesn't warrant being saved for the future unless, of course, we expect that the robot would often need to go from room R1 through door D1 to room R2 to push back the specific box, B1, through door D1 into room R1. We would like to generalize the plan so that it could be free from the specific constants, D1, R1, R2, and B1, and could be used in situations involving arbitrary doors, rooms, and boxes.

In considering possible procedures for generalizing plans we first rejected the naive suggestion of merely replacing each constant in the plan by a parameter. Some of the constants may really need to have specific values in order for the plan to work at all. For example,

consider a modification of our box fetching plan in which the second step of the plan is an operator that only pushes objects from room R2 into room R1. The specific plan might then be

GOTHRU(D1,R1,R2)
SPECIALPUSH(B1).

When we generalize this plan we cannot replace all constants by parameters, since the plan only works when the third argument of GOTHRU is R2. We would want our procedure to recognize this fact and produce the plan

GOTHRU(dx,rx,R2)*
SPECIALPUSH(bx).

Another reason for rejecting the simple replacement of constants by parameters is that there is often more generality readily available in many plans than this simple procedure will extract. For example, the form of our box pushing plan, GOTHRU followed by PUSHTHRU, does not require that the room in which the robot begins be the same room into which the box is pushed. Hence the plan could be generalized as follows:

GOTHRU(dx,rx,ry)
PUSHTHRU(bx,dy,ry,rz)

and be used to go from one room to an adjacent second room and push a box to an adjacent third room.

The plan generalization procedure we have developed overcomes these difficulties by taking into account the internal structure of the plan and the preconditions of each operator. Our first versions of this procedure often introduced irrelevant items in the generalized plan's precondition list. For example, the creation of extraneous parameters might cause the preconditions to include the requirement that box bx be

---

*We use lower case letters to represent parameters.

in room rx and the requirement that box bx also be in room ry. We know that any semantically correct model that satisfies these preconditions will require that rx and ry be instantiated to the same room name; hence, the creation of distinct parameters rx and ry is a superfluous overgeneralization. Such cases of overgeneralization tended to bog down the theorem proving operations during planning and therefore degraded the efficiency of the system. We now eliminate almost all cases of this problem by introducing a processing step after the MACROP is formed that searches for such irrelevancies and removes them by "collapsing" two or more parameters into a single parameter. A complete description of our plan generalizing procedure is given in Ref. (1).

2. Experimental Results

We spent a sizable effort during the year running experiments with the STRIPS-PLANEX system to determine its behavior characteristics. The results of many of these experiments are documented elsewhere, but we will provide summary descriptions of some of them in this section.

Problems were posed to the system in the SRI robot's current experimental environment, which is shown in Figure 1; there are seven rooms, eight doors, and several boxes about two feet high. A typical state of this environment is modeled by STRIPS using about 160 axioms.

a. Operator Descriptions

The operator descriptions given to STRIPS for these experiments model the robot's preprogrammed action routines for moving the robot next to a door in a room, next to a box in a room, to a location in a room, or through a door. There are also operators that model action routines for pushing a box next to another box in a room, to a location in a room, or through a door. In addition, we have included

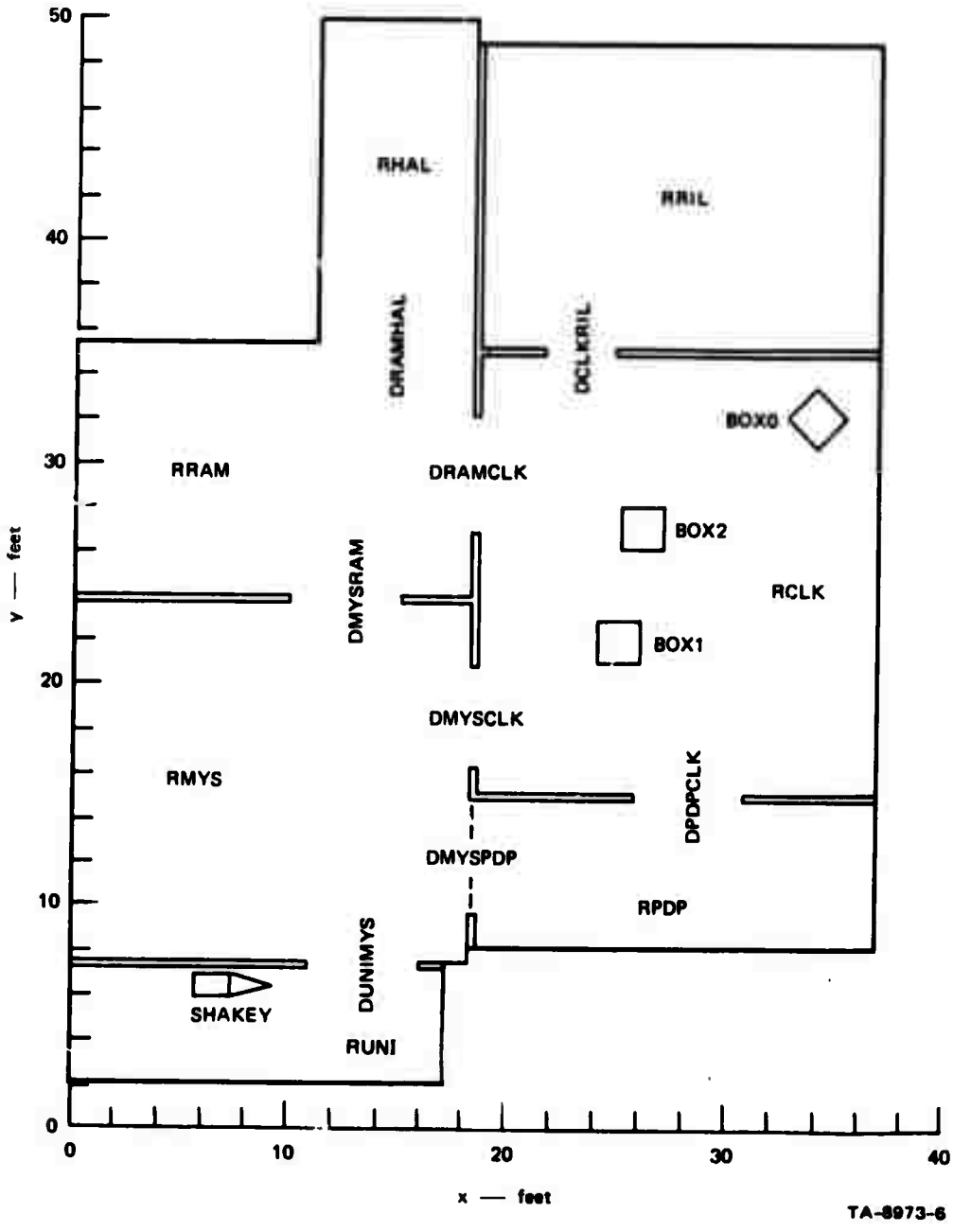


FIGURE 1 MAP OF SHAKEY'S EXPERIMENTAL ENVIRONMENT

operator descriptions that model fictitious action routines for opening and closing doors; these are given in Table 1.

b. Example Problems

A sequence of five problems was designed to illustrate the various ways in which MACROPs are used during planning. We show in Figures 2 through 6 a summary of the system's behavior for each problem in the sequence. Each summary is preceded by a diagram of the problem's initial and final states. STRIPS' attention was directed to the rooms shown in the diagram by closing the doors connecting all other rooms.

Table 2 shows the search tree sizes and running times for the five problems. The problems were run both with and without the use of MACROPs for comparison. Even when MACROPs were not being used for planning we include the MACROP production time, since PLANEX needs the MACROP to monitor plan execution. Note that the times and the search tree sizes are all smaller when MACROPs are used and that the MACROPs allow longer plans to be formed without necessarily incurring an exponential increase in planning time.

c. Further Experiments

In another set of experiments that were run with the new system, the primary goal was to produce long plans. We ran a sequence of eight problems in our robot environment that culminated in the production of a 19-operator plan for fetching three boxes from three different rooms and then pushing the three boxes together. This final MACROP subsumed the seven earlier ones so that only one MACROP was retained by the system. Subsequences of the 19-step MACROP could be used to fetch boxes, push boxes together, move the robot from room to room, and so on.

Table 1

OPERATOR DESCRIPTIONS

GOTOB(bx)	(Go to object bx).
Preconditions:	TYPE(bx,OBJECT),(Erx)[INROOM(bx,rx) ∧ INROOM(ROBOT,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1)
Additions:	*NEXTTO(ROBOT,bx)
GOTOD(dx)	(Go to door dx).
Preconditions:	TYPE(dx,DOOR),(Erx)(Ery)[INROOM(ROBOT,rx) ∧ CONNECTS(dx,rx,ry)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1)
Additions:	*NEXTTO(ROBOT,dx)
GOTOL(x,y)	[(Go to coordinate location (x,y,)].
Preconditions:	(Erx)[INROOM(ROBOT,rx) ∧ LOCINROOM(x,y,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1)
Additions:	*AT(ROBOT,x,y)
PUSHB(bx,by)	(Push bx to object by).
Preconditions:	TYPE(by,OBJECT),PUSHABLE(bx),NEXTTO(ROBOT,bx), (Erx)[INROOM(bx,rx) ∧ INROOM(by,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1),AT(bx,$1,$2),NEXTTO(bx,$1),NEXTTO($1,bx)
Additions:	*NEXTTO(by,bx),*NEXTTO(bx,by),NEXTTO(ROBOT,bx)
PUSHD(bx,dx)	(Push bx to door dx).
Preconditions:	PUSHABLE(bx),TYPE(dx,DOOR),NEXTTO(ROBOT,bx), (Erx)(Ery)[INROOM(bx,rx) ∧ CONNECTS(dx,rx,ry)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1),AT(bx,$1,$2),NEXTTO(bx,$1),NEXTTO($1,bx)
Additions:	*NEXTTO(bx,dx),NEXTTO(ROBOT,bx)
PUSHL(bx,x,y)	[(Push bx to coordinate location (x,y,)].
Preconditions:	PUSHABLE(bx),NEXTTO(ROBOT,bx),(Erx)[INROOM(ROBOT,rx) ∧ LOCINROOM(x,y,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1),AT(bx,$1,$2),NEXTTO(bx,$1),NEXTTO($1,bx)
Additions:	*AT(bx,x,y),NEXTTO(ROBOT,bx)
GOTHRUDR(dx,rx)	(Go through door dx into room rx).
Preconditions:	TYPE(dx,DOOR),STATUS(dx,OPEN),TYPE(rx,ROOM),NEXTTO(ROBOT,dx), (Ery)[INROOM(ROBOT,ry) ∧ CONNECTS(dx,ry,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1),INROOM(ROBOT,$1)
Additions:	*INROOM(ROBOT,rx)
PUSHTHRUDR(bx,dx,rx)	(Push bx through door dx into room rx).
Preconditions:	PUSHABLE(bx),TYPE(dx,DOOR),STATUS(dx,OPEN),TYPE(rx,ROOM),NEXTTO(bx,dx), NEXTTO(ROBOT,bx),(Ery)[INROOM(bx,ry) ∧ CONNECTS(dx,ry,rx)]
Deletions:	AT(ROBOT,$1,$2),NEXTTO(ROBOT,$1),AT(bx,$1,$2),NEXTTO(bx,$1), NEXTTO($1,bx),INROOM(ROBOT,$1),INROOM(bx,$1)
Additions:	*INROOM(bx,rx),INROOM(ROBOT,rx),NEXTTO(ROBOT,bx)
OPEN(dx)	(Open door dx).
Preconditions:	NEXTTO(ROBOT,dx),TYPE(dx,DOOR),STATUS(dx,CLOSED)
Deletions:	STATUS(dx,CLOSED)
Additions:	*STATUS(dx,OPEN)
CLOSE(dx)	(Close door dx).
Preconditions:	NEXTTO(ROBOT,dx),TYPE(dx,DOOR),STATUS(dx,OPEN)
Deletions:	STATUS(dx,OPEN)
Additions:	*STATUS(dx,CLOSED)

*The addition clauses preceded by an asterisk are the primary additions of the operator. When STRIPS searches for a relevant operator it considers only these primary addition clauses.

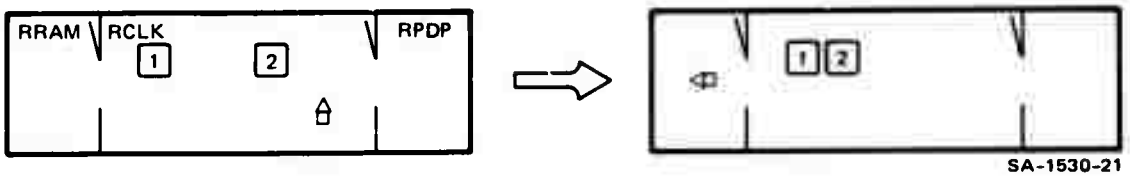


FIGURE 2 EXAMPLE PROBLEM 1

Task Statement:

INROOM(ROBOT,RRAM) ∧ NEXTTO(BOX1,BOX2)

Generalized Plan:

```
MACROP1(par29,par37,par45,par54,par33)
  GOTOB(par29)
  PUSHB(par29,par37)
  GOTOD(par45)
  GOTHRUDR(par45,par54)
```

Comments:

The generalized plan for the first problem in the sequence pushes two boxes together and takes the robot into an adjacent room, given that the robot and the boxes are initially all in the same room.

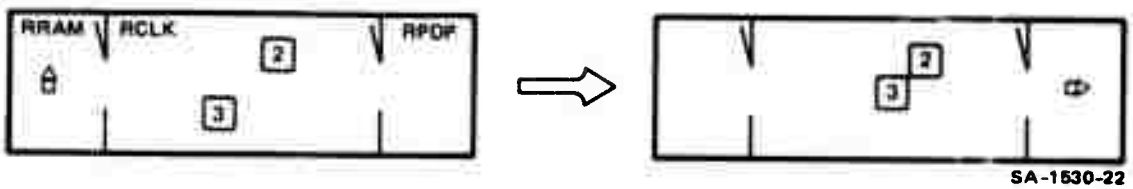


FIGURE 3 EXAMPLE PROBLEM 2

Task Statement:

INROOM(ROBOT,RPDP) ∧ NEXTTO(BOX2,BOX3)

Generalized Plan:

```
MACROP2(par27,par52,par72,par91,par111,par38,par40)
  GOTOD(par27)
  GOTHRUDR(par27,par40)
  GOTOB(par52)
  PUSHB(par52,par72)
  GOTOD(par91)
  GOTHRUDR(par91,par111)
```

Comments:

The second problem is similar to the first except that different rooms and different boxes are used, and the robot begins in a room adjacent to the room containing the boxes. STRIPS uses a tail of MACROP1 to get the robot into the room with the boxes and then uses the entire MACROP1 to complete the plan. The generalized plan takes the robot from one room into an adjacent room, pushes two boxes together in the second room, and then takes the robot into a third room adjacent to the second. The system notes that MACROP1 is completely contained in MACROP2 and therefore erases MACROP1.

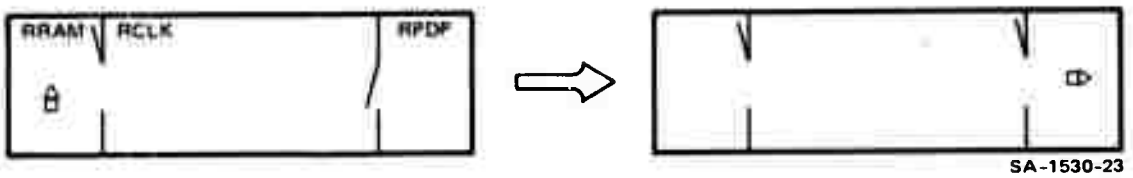


FIGURE 4 EXAMPLE PROBLEM 3

Task Statement:

INROOM(ROBOT,RPDP)

Generalized Plan:

```
MACROP3(par24,par59,par82,par32,par42)
  GOTOD(par24)
  GOTHRUDR(par24,par42)
  GOTOD(par59)
  OPEN(par59)
  GOTHRUDR(par59,par82)
```

Comments:

The third problem entails taking the robot from one room through a second room and into a third room, with the added complication that the door connecting the second and third rooms is closed. STRIPS first decides to use MACROP2 with the box-pushing sequence edited out and then finds that the door must be opened; to get the robot next to the closed door, a head of MACROP2 is selected with the box-pushing sequence again edited out. After formation of the plan to go to the door and open it, the PLANEX scan observes that only the final operator of the first relevant instance of MACROP2 is needed to complete the plan. The generalized plan takes the robot from one room into an adjacent room, then to a closed door in the second room, opens the closed door, and then takes the robot through the opened door into a third.

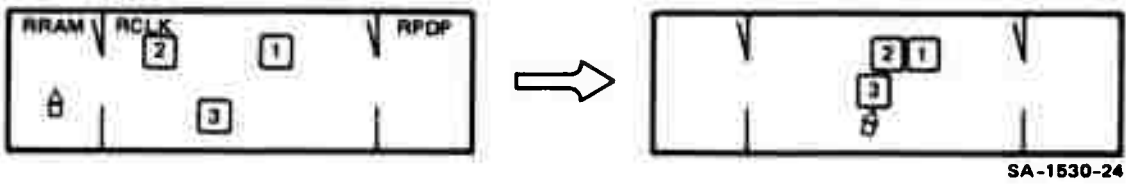


FIGURE 5 EXAMPLE PROBLEM 4

Task Statement:

NEXTTO(BOX1,BOX2) ∧ NEXTTO(BOX2,BOX3)

Generalized Plan:

```
MACROP4(par37,par80,par102,par123,par134,par57,par59)
  GOTOD(par37)
  GOTHRUDR(par37,par59)
  GOTOB(par80)
  PUSHB(par80,par102)
  GOTOB(par123)
  PUSHB(par123,par134)
```

Comments:

The fourth problem requires that three boxes be pushed together, with the robot beginning in a room adjacent to the room containing the boxes. A head of MACROP2 is used to get the robot into the room with the boxes and to push two of them together; the box-pushing sequence of MACROP2 is used to complete the plan, again with the assistance of the PLANEX scan. The generalized plan takes the robot from one room into an adjacent room, pushes one box to a second box, and then pushes a third box to a fourth box.

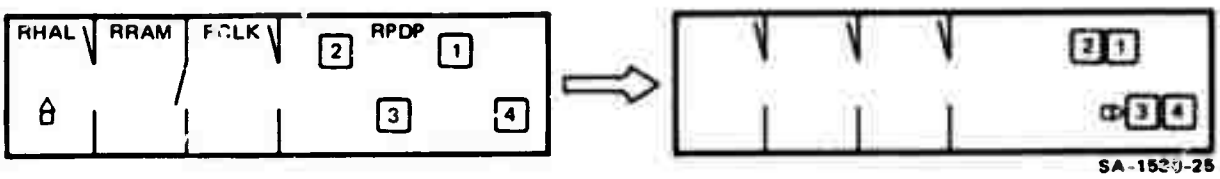


FIGURE 6 EXAMPLE PROBLEM 5

Task Statement:

NEXTTO(BOX1,BOX2) ∧ NEXTTO(BOX3,BOX4)

Generalized Plan:

```
MACROP5(par44,par87,par151,par208,par237,par265,par294,par180,
        par130,par64,par66)
  GOTOD(par44)
  GOTHRUDR(par44,par66)
  GOTOD(par87)
  OPEN(par87)
  GOTHRUDR(par87,par130)
  GOTOD(par151)
  GOTHRUDR(par151,par180)
  GOTOB(par208)
  PUSHB(par208,par237)
  GOTOB(par265)
  PUSHB(par265,par294)
```

Comments:

The fifth problem requires the robot to go from one room into a second room, open a door that leads into a third room, go through the third room into a fourth room, and then push together two pair of boxes. The plan, which is formed by combining all of MACROP4 with all of MACROP3, is well beyond the range of plans producible by STRIPS without the use of MACROPs. Note that although MACROP4 was created by lifting a plan that pushed three boxes together, it has enough generality to handle this form of a four-box problem. Following the creation of MACROP5, MACROP3, and MACROP4 are recognized as redundant and deleted; hence the net result of this learning sequence is to add only MACROP2 and MACROP5 to the system.

Table 2

STATISTICS FOR STRIPS BEHAVIOR
(Minutes)

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
Without MACROPS					
Total time (min:sec)	3:05	9:42	7:03	14:09	--
Time to produce MACROP	1:00	1:28	1:11	1:43	--
Time to find unlifted plan	2:05	8:14	5:52	12:26	--
Total nodes in search tree	10	33	22	51	--
Nodes on solution path	9	13	11	15	--
Operators in plan	4	6	5	7	--
With MACROPS					
Total time (min:sec)	3:05	3:54	6:34	4:37	9:13
Time to produce MACROP	1:00	1:32	1:16	1:37	3:24
Time to find unlifted plan	2:05	2:22	5:18	3	5:49
Total nodes in search tree	10	9	14	9	14
Nodes on solution path	9	9	9	9	14
Operators in plan	4	6	5	6	11

Notes: STRIPS is written in BBN-LISP and runs as compiled code on a PDP-10 computer under the TENEX timesharing system.

STRIPS could not solve Problem 5 without using MACROPs.

The experiments we have been discussing show the use of MACROPs during planning. We have also run experiments with PLANEX to illustrate the use of MACROPs during plan execution. One such experiment is shown in a film[2] that illustrates how PLANEX monitors robot task execution in the seven-room experimental environment. One interesting sequence in this experiment shows the robot attempting to go from one room through a second room into a third room. After entering the second room, the robot discovers that a box is blocking the doorway that leads into the third room. Since PLANEX is working with a generalized plan, the difficulty can be overcome by finding a different instance of the plan that is satisfied. This new instantiation of the plan's parameters causes the robot to be sent from the second room into a fourth room and then into the target third room.

C. Preliminary Specifications for a New Problem Solving System

1. Criteria for the New System

Although STRIPS represents a considerable advance over earlier, theorem-proving based problem-solvers, it still has many shortcomings. On recognizing these, we are faced with the option either of adding some additional features to the STRIPS system or of creating a new system designed specifically to meet certain criteria. We have chosen the latter course, partly because of the availability of the new language, QA4, that simplifies the task of writing problem solving programs of the type we envision.

We would like our new problem solving and execution system to contain as many as possible of the following features.

- Plans with loops and branches

  STRIPS produces "straight-line" code only, although it would not be too difficult to modify it so that it could produce plans with branches.

- Plans incorporating information gathering operators

  The new system should be able to plan to acquire information when it needs it to complete a plan. Such an ability presupposes the ability to generate plans having branches.

- Procedurally defined operators

  The preconditions and effects of STRIPS operators are stated in a rigid format. Operator definitions in our new system will be defined procedurally using QA4 programs.

- Hierarchical planning

  We want our new system to be able to generate a plan at some appropriately high level and then expand the steps of this plan by planning at successively lower and lower levels that encompass more and more details.

- Ability to learn*

  The STRIPS system was able to save generalized versions of the plans it generated so that they could be used later in whole or part as components of new plans. We would like our new system to have this feature also.

- Execution monitoring*

  The STRIPS system was able to monitor the execution of plans in an intelligent manner. We would like the new system to do as well with perhaps a less clear-cut boundary between the planning and execution phases.

---

*The STRIPS-PLANEX system has this feature.

- Compatibility with Speech Understanding Systems

  Ultimately, we would like to be able to interact with the new system through spoken English. We should be able to give it commands and advice, tell it facts, and teach it how to perform new tasks, all by speaking to it.

- Constraints

  The new system should be able to avoid getting into any situation that we define as "illegal."

- Dynamic environments

  We want the new system to perform well in an environment in which other agents of change (e.g., people) are operating.

- Ability to deal with time

  The new planning system should be able to use the concept of time so that it can perform tasks such as "Go to Room 21 at 3:00 p.m."

- Ability to interact with people

  Besides its ability to understand speech, we want the new system to know some simple facts about the people around it, including rudimentary information about their capabilities, locations, and goals. We would want the system to know, for example, whom to ask for advice about the location of some other person.

- Ability to work on conjunctive goals*

  STRIPS had an unsophisticated ability to achieve two or more simultaneous goals. We would like the new system to be able to generate the appropriate plans for highly interdependent goals. (Goals A and B are interdependent if the appropriate plan to achieve A and B entails taking some (but not all) of the steps toward goal A and then taking at least some steps toward B before finishing A.)

For the past few months we have given a great deal of attention to how these features might be achieved. People working on the development of

---

*The STRIPS-PLANEX system has this feature.

the QA4 language have already written some simple illustrative programs that generate plans incorporating loops, branches, information gathering operators, constraints, and conjunctive goals.[4] Mostly these abilities were illustrated one at a time in separate programs, and there are still large questions concerning their synthesis in a single system. The QA4 programs ignored the important question of plan-executing processes and how they interact with plan-generating processes. Nevertheless, work to date makes us feel reasonably optimistic about our ability to program a system containing most of the features mentioned.

We have also worked quite hard on the matter of hierarchical planning. Our ideas on this subject are still tentative, but we think the matter to be important and it is discussed in some detail in the next section.

2. Hierarchical Planning

The ability to plan in a hierarchical fashion has obvious advantages. The general idea is simple: first a plan consisting of a few macro-steps is roughed out in some abstract space. Then the steps of this high level plan are expanded in a little more detail, and so on until the plan is complete at whatever level of detail is defined by the available motor actions. We have explored two somewhat different means of implementing such a hierarchical planner. In the first method, each of the preconditions of each planning operator is given a "criticality number." The criticalities might be assigned initially or they might be computed functions of the predicates and arguments involved. Predicates with high criticality are important even at the highest levels of planning. Thus the precondition PUSHABLE(BOX) in a PUSH(BOX) operator would have high criticality, primarily because the system does not have an operator that can change the value of PUSHABLE. Whether or not a box is pushable is not merely a detail that can be faced at a low planning level.

Predicates with low criticality can be ignored at the highest levels of planning. Whether they are true or false is immaterial since they can (presumably) be easily given the desired value at a lower planning level. Thus at the highest planning level the predicate dealing with the robot's heading, say, can be ignored since it can easily be set correctly by lower levels.

Using these criticality numbers, we can generate plans hierarchically in a straightforward manner. First the planner generates a plan with a high criticality threshold. That is, only those preconditions whose criticality exceeds a rather high level are considered. Such a plan consists of a sequence of operators each of which has rather weak preconditions. Next we could, say, pick the first operator in the sequence (whose weak preconditions are satisfied in the initial model) and lower the criticality threshold on its preconditions. Some more planning steps may now be necessary to satisfy the somewhat strengthened preconditions. In this manner we generate a plan that gradually considers more details until finally the preconditions with the lowest criticality numbers are also considered.

If at any stage the preconditions of an operator in the plan cannot be satisfied at a lower criticality threshold, the plan at the next highest level is rejected and an alternative must be found.

There are important questions here regarding the order in which steps in a plan at a given level ought to be expanded to lower levels. An interesting special case might be called FIFE (first in, first expanded). In the FIFE mode, the first step in the high level plan would be expanded at the next level of detail. Then the first step in this expansion would be expanded, and so on until finally the first step corresponded to an executable action. Here we face a choice. Do we execute this operator and then continue with FIFE or do we continue expanding some or all of the rest of the plan before any executions are allowed?

Another important question concerns the level at which new planning is to be done as we gradually lower the criticality threshold. Suppose, for example, that the criticality threshold is at some intermediate level and we are testing the preconditions of one of the operators in the plan. If one of these preconditions is not satisfied, should we generate a high level plan to satisfy it or should we generate a plan of a level corresponding to the current setting of the criticality threshold? Our current opinion is that any new planning activity always ought to occur at the highest level. More details about this particular method are contained in a memo by Earl Sacerdoti.[5] This technique is now being implemented as an addition to the STRIPS-PLANEX system.

A second method by which hierarchical planning can be accomplished entails writing separate operators for different planning levels. For example, we might have the following hierarchy of operators for achieving the predicate INROOM(ROBOT,ROOM): GOTORM1, GOTORM2, and GOTORM3. The highest level operator, GOTORM1, would have rather mild preconditions, such as, say, a test for the existence of the target room. The next operation, GOTORM2 would have somewhat more restrictive preconditions, say that the robot must be in a room adjacent to the target room. Finally GOTORM3 might insist that the door between the target and adjacent rooms must be open.

First a plan would be generated using only high level operations. Then each of the operators in this plan beginning, say, with the first would be replaced by its next lower level operator and plans (at the highest level) would be generated to achieve its preconditions, and so on. This procedure would work very much like the one using criticality numbers, except that there is now no necessity that the preconditions of a high level operator be a subset of those at lower levels.

There are many details to be worked out before we can begin implementing a hierarchical planner. We haven't quite decided yet, for example, how to administer the search process in generating hierarchical plans. We desire an executive that can resume work on a possibly incomplete plan at a higher level should some step in a lower level plan run into trouble. Communication between levels is also a problem. We might desire that information discovered by a lower level planner be available for use in generating alternative high level plans.

Perhaps the most difficult of our tasks will be to integrate a hierarchical planning feature successfully with all the other features we mentioned in the last section. In particular, we have had some difficulty in deciding how a hierarchical planner and a plan execution system ought to interact.

D. Multirobot Experiments

In parallel with our work on a new problem solving system, we are planning some experiments using STRIPS and the present robot vehicle, SHAKEY, in conjunction with a Unimate arm. (A Unimate is a commercially available, fixed, industrial manipulator that is being used in our laboratory for experiments in the application of techniques in artificial intelligence to industrial automation.) Use of both SHAKEY and the Unimate will allow us to explore some interesting problems concerned with multirobot cooperation. Three fundamental types of experiments are envisioned:

(1) Those in which each robot is viewed by the computer as an independent motor device exclusively under its control.

(2) Those in which each robot is strictly autonomous and operates effectively in parallel by time sharing the same computer facility.

(3) Those which are a mixture of the first two in which one robot plays the roll of master and the other slave.

Type (1) experiments would not require the duplication of any robot software on the computer, but merely the addition of a new set of operators for the Unimate. The only aspect of robot communication protocol that is qualitatively different from the one-robot case is the fact that the two devices can operate asynchronously and thereby possibly physically interfere with one another, if careful attention is not paid to the sequence and timing with which operations are carried out.

Our interest in types (2) and (3) experiments stems in part from our desire to begin considering worlds containing large numbers of robot devices. In such a world, it would not be reasonable to assume that all robots were controlled by a single computer. Instead, we would assume autonomous robots that might be called together by a human to perform a task. The human would not, presumably, want to specify how the task is to be subdivided among the robots; the robots must figure this out among themselves.

As a matter of experimental convenience, we will use only two robots (Shakey and the Unimate) and a single computer. Hence, our type (2) experiments require the duplication of certain subroutines and model structures in computer memory to allow each robot to maintain its identity. Provision must be made in each robot's model for the current state, goals, and potential capability of its counterpart, including the ability to communicate. By definition, neither robot can have direct access to the other's model data; each must carry out a symmetric dialog to discover the other robot's intentions. The distinction between "knowledge" and "belief" is now important, in contrast with type (3) experiments in which the master robot is assumed by definition to know precisely what the slave knows.

A possible experiment might be for Shakey and the Unimate to be given a joint task of turning a box upside down and moving it to a distant location. This task would capitalize on the unique ability of the Unimate to turn boxes upside down and the unique ability of Shakey to move them around. In a type (2) experiment, both robots would need to communicate with one another according to some fixed protocol in order to develop jointly a common plan of action to accomplish the goal. This simple type of experiment can be easily embellished by adding disjoint (possibly conflicting) parallel goals for each robot. Under more stringent constraints the difficulty in establishing and maintaining a dialog can be made arbitrarily complex.

III PERCEPTION

A. Introduction

In the past year we have formulated and refined a new approach to machine perception intended to overcome the major limitations of existing systems and to achieve useful real-world perceptual capabilities. This section presents, in considerable detail, both our design for a system for analyzing perceptual data and a research methodology for implementing the system.

The goal of most vision research has been to describe simple geometric environments in an exhaustive, bottom-up fashion. Unfortunately, many crucial perceptual issues--such as information overload, generality of perceptual strategies, suitable representations for real-world objects, and segmentation of textured objects from the background--do not arise in this problem domain. Consequently, that goal has proved largely self defeating, leading to strategies and systems that could not be extended to cope with richer environments. For example, the absence of natural perceptual redundancy and context meant that each object could only be recognized in terms of a completely articulated boundary shape description. Such descriptions, while useful, are difficult to obtain for many real world objects. Moreoever, detailed shape is often not the most appropriate distinguishing characteristic. The emphasis on shape also demanded unreasonable sensitivity and reliability from the initial boundary extraction routines. This undirected sensitivity further limited the systems to textureless objects and background in order to avoid overwhelming the scene analysis stage with irrelevant edge detail.

1. Perceptual Philosophy

In contrast to the above, we have chosen as our primary research objective the task of finding specified objects in complex real-world environments. This objective is consistent with the information requirements of our robot, which typically is concerned only with locating particular objects involved in its current task.

We proceed on the premise that there exist easy ways of "seeing" things. The redundancy of visual cues and contextual constraints allows a desired object to be distinguished from others on the basis of a small subset of the available features. To illustrate, a human can usually be distinguished from other contents of an office simply as a moving blob whose surface area is larger than two square feet.

Moreover, even when such complicated features as shape are required, then if the context is suitably limited, a simple distinguishing measure related to that feature may suffice to resolve remaining recognition ambiguities. Thus, either the relative position of the center of gravity or the presence of sharp corners should be sufficient to distinguish the shape of a person from that of Shakey.

2. Design Considerations

The above point of view suggests the following design considerations for a perceptual system.

a. Perception as Problem Solving

Perception should be construed as a problem solving process; the system must utilize its knowledge of the current real-world environment, and of its own perceptual capabilities to plan where and how to look for a specified object. Specifically, the computer must use its knowledge to select features of the desired object that are both distinguishing and easy to see.

The utility of such 'distinguishing features' is critically dependent on what is known at the current stage of analysis. Hence, it is unreasonable to preprogram these recognition strategies, except in the simplest and most static environments. Moreover, a system that can plan its own strategy has inherent generality; it should be able to function in any environment for which its knowledge base and perceptual primitives are adequate.

b. Sequential Decision Paradigm

It is usually unnecessary to examine all features of an object to arrive at a confident recognition hypothesis. Perception should proceed like the game of 20 questions. Simple descriptive attributes (e.g., color, motion, size) should be used initially to establish a limited context. Remaining ambiguities can then be resolved within this context, using the distinguishing components of complex attributes (e.g., shape, texture).

c. Representation of Complex Objects

Representations for describing the shape and texture of complex real world objects are not yet available. Indeed, symbolic descriptions may not even be feasible. However objects can be represented, for purposes of discrimination, in terms of the many crude measures of shape (e.g., perimeter-squared/area, length/width of bounding rectangle) and texture (e.g., statistics, power spectrum) that are available. The system should use the simplest representations sufficient to distinguish the object of interest in a given context.

d. Multiple Sensors

The likelihood of finding suitable surface attributes for distinguishing a given object increases with the number of independent

sensory modalities. Furthermore, simple discrimination in each of several sensory modalities should be a cheaper, more reliable alternative to using more detailed descriptions in a single modality; color and surface orientation (obtained directly from range data) are substantially easier to process than shape and texture, the principle means of distinguishing objects in gray-scale images.

e. Goal Directed Feature Extraction

A key problem in doing perception by distinguishing features is to extract reliably the features. One lesson that has been repeatedly learned in a decade of vision research is that feature extraction cannot be adequately performed bottom-up, as a preliminary to a knowledge-based interpretative process.

Rather, the system must be integrated so that recognition strategies are based on knowledge of which features are easy to extract in a given environmental context. The low level routines should then concentrate on extracting those specific features, guided again by knowledge of their distinguishing attributes. For example, regions can be grown on the basis of surface attributes that are known to be both homogeneous over the goal object and distinguished from those of other nearby surfaces, previously found or anticipated in the environment.

f. Incremental Acquisition of Knowledge

The substantial amount of ad hoc world knowledge required to plan perceptual strategies is most reasonably acquired in an incremental fashion. The system should thus be designed to request additional information from a user at times of failure, indecision, or on encountering a new object, and to incorporate this information immediately in a revised strategy. The new strategy establishes empirically whether the

current description of an object is sufficient to distinguish it from others already known.

3. Organization of this Section

We have completed the conceptual design of a perceptual system, incorporating each of the features listed above.

In Section III-B we present an overview of the perceptual strategy.

Section III-C describes a system design, modeled after utility theory, for planning and executing this strategy.

Section III-D outlines our planned approach for utilizing high level world knowledge to direct low level feature extraction.

Section III-E summarizes related work on interpreting color and range data, to realize a multisensory capability.

Section III-F concludes with our plans for implementing and experimenting with the system.

B. Overview of System

1. Scenario

Before describing the system, let us first describe a scenario that has played an important role in helping refine our thinking. Its inclusion here serves three functions: it illustrates the intended mode of system operation, it provides explicit examples to clarify later discussions on perceptual strategy, and it establishes some concrete operational objectives against which our system, when implemented, will be evaluated.

a. Scenario Objective

The scenario objective is to analyze an uncluttered office environment like that shown in Figure 7, using brightness, color, and range as sensory inputs. The basic task is to find a designated object (e.g., a chair). The system will designate its comprehension by outlining the indicated object on a grey-scale display of the scene. A second task is to describe the scene, which in this limited environment can be accomplished by commanding the system to find each of the small set of known objects.

b. Knowledge Base

The system's world model will initially contain five typical objects selected from offices, and three room fixtures. The objects are semantically constrained to appear in normal office relationships (e.g., chairs are not allowed to be on tables). Table 3 conveys, informally, the scope of information that will be available to the system. The given attributes and relations do not constitute complete descriptions, but should be adequate to distinguish among the objects using color and range data. (Partial descriptions are, in fact, preferred because they provide possibilities for generalization. The descriptions can always be refined interactively, should the system err on their account.) The descriptions also indicate the range of specificity with which knowledge can be provided. Thus, dimensions may be given exactly, corresponding to particular objects (e.g., the door and chair in Table 3), or with a tolerance encompassing a class of possible objects (e.g., the picture in Table 3).

Though spatial relationships are expressed here as gross symbolic constraints (e.g., back of chair "parallel to and above" seat), it is often more convenient, in practice, to represent such metrical and

FIGURE 7 A SIMPLIFIED OFFICE ENVIRONMENT REPRESENTING OUR EXPERIMENTAL DOMAIN

Table 3

INITIAL WORLD MODEL

Object	Attributes	Relations
Wall	Vertical plane Dimensions: > 7 ft high (except at door) and > 8 ft long Shape: rectangular, except for doorway Color: buff, homogeneous	Adjacent and perpendicular to floor (below) and other walls (side). Adjacent to doorway and door.
Floor	Horizontal plane (height = 0) Shape: linear boundaries Dimensions: at least 5 ft of extent along x and y axes, delimited by intersection with walls. Color: buff with white and brown streaks	Adjacent and perpendicular to walls, door, and doorway. Continuous through doorway. Supports chair, table, basket.
Door		
Door body	Vertical rectangular prism Size: 3-1/2 ft wide by 7 ft high by 1-1/2 in. thick Color: brown, wood grain	Hinged (-90 < angle < 90) to wall on one vertical edge. Adjacent to floor on bottom.
Door knob	Cylindrical prism Dimensions: 2 in. (diameter) by 2 in. (length) Color: silver	On (i.e., base coplanar with) both wide faces of door body at height = 38 in. It is 5 in. from unhinged edge.
Chair		
Seat	Horizontal rectangular plane prism Dimensions: 18 in. by 18 in. (horizontal plane, top) and 18 in. by 4 in. (vertical side planes)	Supported by legs which are attached at each corner of the bottom rectangular face.
	Height: 16 in. to center of gravity Color: tan or gray	Width of back parallel to and above edge of seat.
Back	Vertical rectangular prism Height: 24 in. to center of gravity Dimensions: 12 in. high by 17 in. wide by 1-1/2 in. thick Color: same as seat	

Table 3 (concluded)

Object	Attributes	Relations
Chair		
Legs	Vertical rectangular prisms Number: 4 Color: gray or cocoa Size: 1-1/2 by 1-1/2 by 14 in. (height)	Supported by floor.
Table		
Top	Horizontal rectangular prism Color: buff or gray Dimensions: 3 to 6 ft long, 2 to 4 ft wide, and 28 to 36 in. high	Supported by table legs.
Legs	Vertical rectangular prism Height: 20 to 36 in.	Supports table top at corners. Legs supported by floor.
Basket	Vertical cylinder Color: gray or brown Hollow on top Dimensions: 13 in. (diameter) by 14 in. (height)	Supported by floor.
Picture	Vertical rectangular plane Dimensions: 8 in. to 30 in. (length or height) Multiple colors, usually in small regions.	On (coplanar with) wall.
Telephone		
Case	Horizontal rectangular prism Dimensions of base: 5 in. wide, 8 in. long, and 1 in. high Color: black	Supported by table.
	Horizontal rectangular wedge Dimensions: 5 in. wide, 8 in. long, and 3-1/2 in. high Color: black	Supported by and aligned with top of rectangular prism.
Dial	Cylindrical prism with multiple holes in end Dimensions: 3 in. in diameter by 1/8 in. thick Color: gray	Centered on sloping face of wedge.

topological relationships implicitly, in the form of structural models (like those used for computer graphics). In our initial implementation we plan to use crude structural models for objects with several parts. A chair, for example, will be represented by rectangular prisms for seat, back, and legs. Such a model should convey spatial relationships in sufficient detail for planning perceptual strategies, but be simple to obtain and use in early experimentation.

2. Perceptual Strategy

The search for an object proceeds in two phases, called acquisition and validation. During acquisition, the multisensory image is sampled for characteristic surface attributes of the desired object. If a sample satisfying all criteria is found, a sequence of top-down validation tests determines whether the acquired sample does, in fact, belong to the desired object, or to another object with similar surface characteristics. Each of these search phases will now be discussed in more detail.

a. Acquisition

The selection of acquisition attributes for sampling is based on such considerations as:

(1) Criteriality--the attribute should invariably be associated with the object (e.g., the floor is always horizontal).

(2) Distinguishability--the attribute should not also be characteristic of other objects expected in that context.

(3) Measurability--the attribute should be reliably obtained from simple, localized processing of sensory data.

These criteria are applied to the characteristics of each surface of an object to ascertain the best set of surface attributes. Let us illustrate our approach with the goal of finding a chair. The initial system implementation will include primitives for testing height, color, and local surface orientation. The discussion on multisensory data interpretation in Section III-E suggests that height will be the most easily measured attribute, followed by horizontal orientation, color, and vertical orientation. Refer now to the description of a chair in the basic world model (Table 3). Height and horizontal orientation are both criterial to the seat of a chair, and together are unique in that environment. The color alternatives make that attribute less criterial. Since vertical orientation, the primary attribute of the back of the chair, is both less distinguishing and more difficult to test, a chair should be sought by sampling for a height of 18 inches, and checking successful points for local horizontal orientation.

Table 4 summarizes the anticipated best acquisition attributes for each object in the scenario. If a search fails to satisfy one of these criteria, the system can then select alternative attributes based on plausible explanations for the failure. For example, the seat of a chair might not be visible if the chair were viewed from behind. Thus, failing to find a horizontal sample of appropriate height for a seat, the system could next look for characteristic attributes of a back support.

Sampling may be localized to specific areas of the scene on the basis of objects already recognized. For instance, a wastebasket need only be sought in areas bordering the floor region. In fact, the planning algorithm discussed later will have the option of looking first for a contextually related object that is larger or other ise easier to find, in order to localize the desired object with reduced total search

Table 4

ATTRIBUTES AND CONTEXTURAL RELATIONS FOR ACQUISITION

Object	Acquisition Attributes (Test in order shown)
Wall	Color - buff; orientation - vertical
Floor	Height ≈ 0 in.; orientation - horizontal
Door	Color - brown; orientation - vertical
Chair	Height ≈ 18 in.; orientation - horizontal; color - tan or gray
Table	Height 20 to 36 in.; orientation - horizontal; color - buff or gray
Basket	Color - gray or brown; orientation - vertical; context - on floor
Picture	Buff/nonbuff boundary; context - on wall
Telephone	Color - black; orientation - vertical or inclined; context - on table

effort. Thus, to find a telephone, it might pay first to find a desk. Humans often pursue a similar style of perceptual search.

b. Validation

The validation process begins by checking the global attributes of the surface surrounding the acquisition sample. This surface is extracted by grouping the acquisition sample with proximate samples having similar attributes. The resulting region is then checked for appropriate size, shape, global uniformity and so on.

The global attributes help distinguish the desired surface from surfaces with similar acquisition attributes, belonging to other objects in the knowledge base. We intend to rely on the consensus of

several crude, individually tolerant tests related to these attributes, rather than actually to extract a detailed description of any particular one. For example, the ratio of perimeter squared to area, or of length to width, might suffice as a representation of shape. The system will select the simplest of such tests that provide adequate discrimination. In addition, certain critical dimensions (e.g., length, width, area) will be measured absolutely (using range data) to reduce the likelihood of an unknown object slipping through the explicit discriminations.

Unfortunately, past experience suggests that a test is as likely to fail because of errors in region growing as because the wrong object was acquired. We hope to overcome the unreliabilities inherent in the region growing process by tightly controlling it with feedback from the evaluation objectives. This will be discussed in section III-D.

If the confidence remains indecisive after surface evaluation, or if specific ambiguities remain, then validation continues by seeking additional parts of the object (e.g., the back of a chair, the knob of a door). These subobjects can themselves be acquired and validated by using the basic programs recursively.

Additional confidence can be obtained by finding other objects in appropriate contextual relationships, such as, "support" (e.g., telephone on table), adjacency (e.g., wall and door), functional (e.g., hammer near nails), and the like.

Validation proceeds as a sequential decision process that terminates when a definitive level of confidence has been reached. After each feature is sought a decision must be made whether to accept the original acquisition hypothesis, to reject it (and resume sampling) or to continue the validation process. This decision will depend on parameters of required confidence and allocated budget, reflecting the global

importance of this current goal to a higher level robot strategist. The decision to terminate validation also depends, in part, on the degree to which the desired object must be localized in the scene, since the more features of an object that have been found, the more tightly location and orientation are constrained.

C. Conceptual Design

1. Introduction

We would now like to describe in detail a system designed to plan and execute the kind of perceptual strategy described above. The design goal is to find a specified object with minimal cost (i.e., computation time) within an allotted budget, while maintaining a required level of reliability (i.e., effectiveness). This goal requires that the system utilize all available information about the object, the general environment, detector routines, and current sensory information.

The system will generate a planning tree representing alternative ways to acquire and then validate an object. Initially, only paths emanating from the most promising acquisition features will be explored in detail. Moreover, less detailed cost and reliability estimates will be used in calculating the utility of validation features, to simplify planning. The system will then proceed to execute the most promising path based on its initial information, planning in greater detail as the strategy successfully progresses to the validation phase. On the other hand, if a strategy fails, the system can utilize the new information it acquired during execution to choose an alternative execution path or perhaps to resume planning a previously unpromising approach. Since knowledge is constantly acquired during the perceptual process, planning and execution will be tightly interwoven.

The planning/execution interaction will be coordinated by the GOAL MONITOR (GM). The task of the GM will be to account for the effects of the success or failure of one subgoal on the rest of the subgoals in the planning tree awaiting execution. Thus, achieving one subgoal could remove others from consideration, could restrict the search space for others, or might change the importance of satisfying them. The effect of goal monitoring is to use the most current information available to dynamically coordinate the planning and execution. This process retains the flavor of utility theory (i.e., always do whatever is currently most promising), but many practical drawbacks of the formal theory can be avoided. In particular, the combinatorial difficulties entailed in global optimization will be minimized by considering many processes to be independent until much later in the planning and execution sequence. Since we do not have accurate utility estimates, we cannot (and do not wish to) look too far ahead in our planning.

The satisfaction of our goal proceeds in three main steps:

(1) The planning phase, which may include the generation of alternative paths to the goal. For example, to find a telephone either look for the telephone directly, or look for a desk first and then look for a telephone on the desk. The initial planning will be concerned with acquisition of the object. Additional considerations, such as validation and refinement, will then be examined for the most promising paths.

(2) The acquisition phase, where some features of the current best subgoal are used to attempt to locate it in the scene. For example, if it is determined that the best way to find the telephone is via the desk, it may be decided to attempt to acquire the desk on the basis of a large surface of a certain color.

(3) The validation phase where, having found the initial acquisition feature, the system attempts to verify the acquisition through other parts or contextual

relationships. For example, having located what appears to be the top of the desk, the system may attempt to show that it really is the desk by finding the sides.

2. Definitions

Before proceeding, we would like to define certain parameters and explain how we plan to use them. We will assume that the input to the system is a list of goals of the find type and a budget allocation. Associated with each goal will be the interest (I) that the robot executive has assigned. The interest is a number from 0 to 1, and is a measure of the importance of the goal to the executive. The I values will be used to set the confidence with which the goal must be achieved. From the set of I values, the system will compute the normalized interest (NI) for each goal. The NI of a goal will be the I of that goal divided by the sum of the I's for all goals.

The budget is split into two parts: a fixed percentage is to be used for planning, and the rest for execution. The allocation of planning budget will be made in proportion to the NI of each goal. This ensures that the effort expended on a goal is proportional to its overall interest to the system. After planning for a goal, any remaining budget for that goal is reallocated among waiting goals. The execution budget allocation will be based on the results of the planning, and will change dynamically during the course of execution and replanning.

We will take cost to be a measure of the processing time required for completion of a task to a certain level of reliability. The reliability is the probability that the results returned for a given task are correct--we will use the term confidence synonomously with reliability.

We will use FOM to represent a computed figure-of-merit for each node in the planning tree. This will take into account such things as the expected cost of achieving that subgoal with a required confidence, and its importance to higher goals. We will distinguish two factors used in computing the FOM: a direct FOM (DFOM) and an indirect FOM (IFOM). The DFOM measures the ease with which a feature (a tip node of the tree) can be acquired directly--that is, by looking for the feature in a picture. The IFOM measures the utility of the contextural information provided by one object to the goal of locating another object. The IFOM is a property of a relation between two objects (and is therefore associated with a branch of the planning tree). It gives an estimate of how close you are to satisfying a higher level goal given satisfaction of a lower one. The FOM of an intermediate node of the tree will be generated by backing-up the DFOMs and IFOMs of the tips and branches of the tree beneath it. The values of the IFOMs, DFOMs, and FOMs will all be within the interval [0,1]. The FOM of a goal will be used to decide which is the best thing to pursue next during planning and execution.

We will also be referring to three data bases. The first is the VISUAL MEMORY (VM), which will contain permanent perceptual knowledge about objects, parts of objects, visual features of parts, and relationships between them. The information in VM will be obtained mainly by interacting with a human operator when new objects are encountered or when a strategy based on existing knowledge fails. Some details of this interaction appear in Section III-F.

The second data base is a short term world model (STWM) in which the results of current processing are represented as instances of the items in VM. An accompanying coordinate transformation gives the actual location and orientation of each instantiation. There is also a low level data structure for representing regions, boundaries, and similar partially digested information about the scene.

The third major data base is the detector library, containing both perceptual operators and knowledge about their application. This knowledge includes estimates of cost and reliability, as well as pragmatic advice on the use of given operators in particular contexts.

3. Planning

We would like to give an overview of the initial stage of planning by describing in general the course of planning. For the moment we will assume the existence of values of DFOM and IFOM at each stage, later discussing criteria that may be used in their computation.

a. Plan Generation

First we need to describe the hierarchical structure of visual memory, illustrated in Figure 8. Objects have parts, and parts

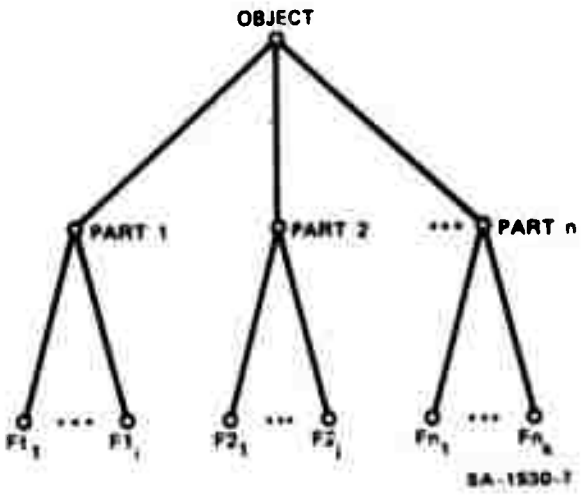


FIGURE 8 VISUAL MEMORY HIERARCHY

have features. Features are the visual attributes that can be found directly in an image. Examples of features include color, surface orientation, texture, size, and shape. The basic concept of "part" is

a surface with a collection of relatively homogeneous features. (Parts may themselves have parts, but we will not consider that here.)

Let us assume that we are trying to satsify a single goal, FIND OBJECT1, and have already allocated part of the budget to that goal for planning. Let us also assume that we have an FOM value for the object to be found that we will use as a cutoff value in planning (the target FOM). We will evaluate the advisability of attempting to find an object directly versus finding it indirectly at each stage of the plan. The plan we ultimately devise will be the "best" branch of the planning tree.

The first level of the planning tree for finding OBJECT1, is merely the object-feature hierarchy detailed above, and is illustrated in Figure 9. OBJECT1 has two parts, P1 and P2. In turn, P1 has features

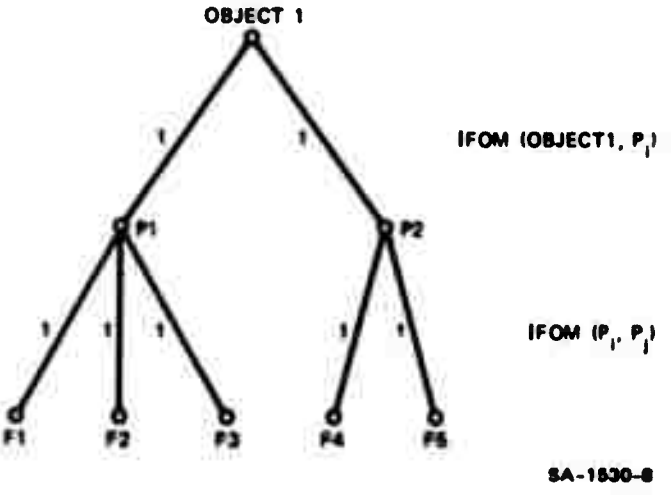


FIGURE 9 PLANNING TREE FOR FINDING OBJECT 1

F1, F2, and F3. P2 has F4 and F5 for its features. The IFOM (OBJECT, PART) is always 1, since finding a part of an object brings you all the way to that object. Likewise, the IFOM (PART, FEATURE) is also 1. We will write the IFOM values on the branches (see Figure 9).

The FOM of a tip node will be the DFOM of the feature at that node. The backed-up FOM value of a nontip node will be computed from the FOMs of the nodes directly below it in the tree, and the IFOMs of these nodes with respect to it, by taking the maximum of the set of values of IFOM(PARENT,OFFSPRING) * FOM(OFFSPRING). The FOM of a node can therefore be considered to be the FOM of the planning tree from that node. Thus, the current FOM(OBJECT1) is equal to

$$\max\{\text{FOM(P1)} * \text{IFOM(OBJECT1,P1)} , \text{FOM(P2)} * \text{IFOM(OBJECT1,P2)}\} .$$

Now, we have

$$\text{FOM(P1)} = \max\{\text{FOM(F1)} * \text{IFOM(P1,F1)} , \text{FOM(F2)} * \text{IFOM(P1,P2)} , \text{FOM(F3)} * \text{IFOM(P1,F3)}\}$$

or, since each IFOM is equal to 1,

$$\text{FOM(P1)} = \max\{\text{DFOM(F1)} , \text{DFOM(F2)} , \text{DFOM(F3)}\}$$

and similarly

$$\text{FOM(P2)} = \max\{\text{DFOM(F4)} , \text{DFOM(F5)}\} .$$

Therefore,

$$\text{FOM(OBJECT1)} = \max\{\text{DFOM(F1)} , \text{DFOM(F2)},\ldots, \text{DFOM(F5)}\}^{*} .$$

Suppose now that the FOM(OBJECT1) is not high enough (i.e., it is not greater than the target FOM value), and suppose also that we have planning budget left. We would like to consider looking for a related object to help us find OBJECT1. We wish to consider IFOM(OBJECT1, obj) for all objects, obj contexturally related to OBJECT1. We can immediately reject any object, obj, for which IFOM(OBJECT1,obj) is less than the current FOM(OBJECT1), since even if FOM(obj) is 1 (the maximum

---

*Those readers familiar with probability theory may recognize some similarities between the FOM computations and the calculation of probability of two dependent events.

possible), it cannot improve FOM(OBJECT1). (In practice, we will consider objects whose IFOM with respect to the original object are almost as high as the FOM of the original, since these estimated utilities are likely to change somewhat during execution.

Now, suppose two objects, OBJECT2 and OBJECT3, satisfy the test on IFOM. We grow the tree shown in Figure 10, and allocate

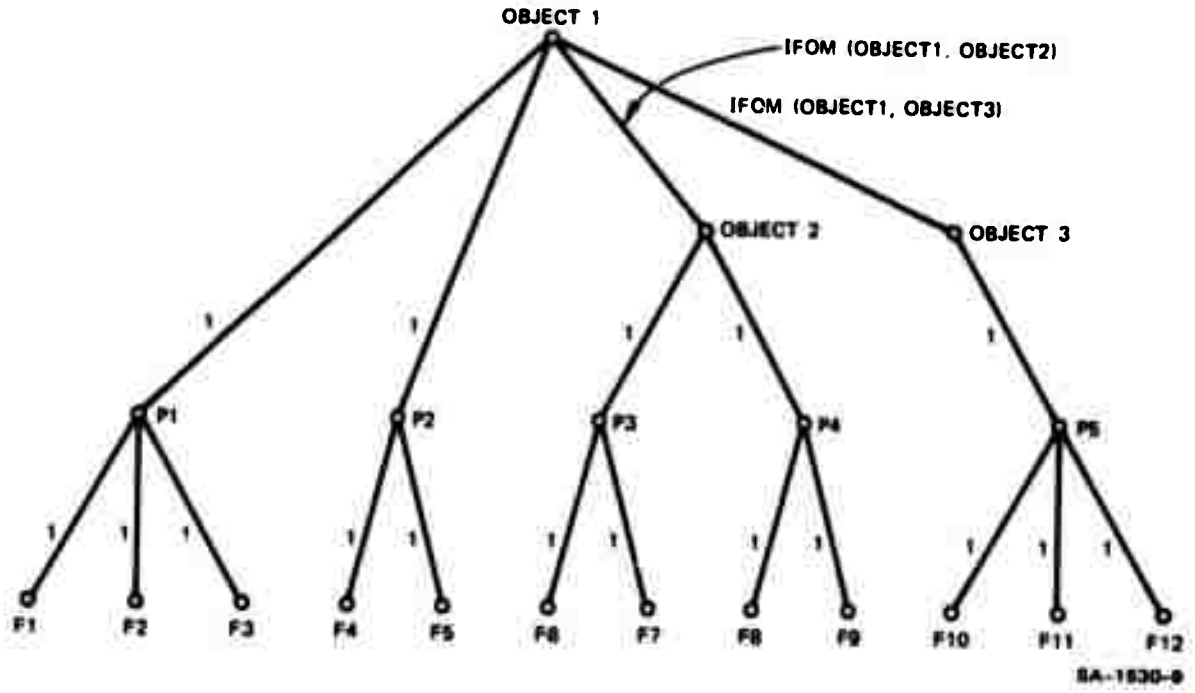


FIGURE 10 EXTENDED PLANNING TREE FOR OBJECT 1

a percentage of the planning budget remaining, based on the relative IFOM values, to each of the two new objects. Using FOM'(OBJECT1) as the new FOM value, we have

$$\begin{aligned} FOM'(OBJECT1) = \max\{&FOM(OBJECT1), \\ &IFOM(OBJECT1,OBJECT2) * FOM(OBJECT2) , \\ &IFOM(OBJECT1,OBJECT3) * FOM(OBJECT3)\} \end{aligned}$$

with FOM(OBJECT2) and FOM(OBJECT3) calculated as above.

As each new node is added to the tree, a backward pointer to its parent is stored with it. These pointers will be used by the GM during execution to recalculate FOMs based on success or failure of subgoals.

Whenever we start planning to find an object, we insert a pointer to the corresponding branch of the planning tree on the property list of that object in VM. Accordingly, we do not have to plan for finding an object more than once. Whenever we reach an object in the course of planning, we increment a counter associated with that object. When we are finished, we have a count of the number of times an object has appeared in the plan. We will use this count in the execution of the plan in order to give higher weight to subgoals that can help achieve several goals.

This mechanism allows a rudimentary form of learning. Previously successful plans would remain on the objects property list in VM with an accurate FOM value and be available for subsequent use. The system will be designed to always check each node for an existing strategy before planning anew. If a plan is available, the FOM stored with that plan is used in computing FOMs for higher nodes. After checking the subgoals in the plan against other subgoals, we will treat the new branch of the tree just like any other. Property list pointers are also a convenient way to incorporate experimental, manually generated strategies.

We continue the planning process until one of several possible termination criteria is met: Either we run out of budget, there is no way to improve our current FOM, or the current FOM is good enough. Whenever we "terminate" a planning branch, we really only suspend it. Thus, if we decide later that perhaps it should have been considered further, we can start from where planning was discontinued. This will be important should we terminate a subgoal because of budgetary limits, and then return to that subgoal along another branch of the tree when

we have more budget to allocate to it. The process ends returning the best FOM calculated and the plan that produced it. We will repeat this process for any remaining goals on our list.

At this point we have produced a plan for finding an object based on information on hand, and some fairly crude estimates of DFOM and IFOM. We have computed an FOM for the desired goal based on the apparent best choice of features for acquisition at all nodes. If there is not a clear best choice, we can continue evaluation of the most promising branches, using somewhat more sophisticated (and expensive) criteria. We will describe this in detail later.

b. Criteria for FOM

We would like to describe some of the criteria to be used in the computation of the figures of merit, DFOM, and IFOM. The IFOM is, in essence, a measure of how spatially constraining one object is on another. It will take into account the physical relationship of one object to another, the probability with which this relationship holds, and the amount of effort required to localize the constrained object given the constraining one. For the parameter to be meaningful, the computation must take into account specific information available in the VM. For example, we know that the wall is always adjacent to the floor, and that given the wall's location, the amount of work required in finding the floor is not very great--just scan down the wall looking for a certain kind of discontinuity. This means that IFOM(FLOOR,WALL) will be fairly high. Since the floor is not always adjacent to the wall (there may be doors), and since it may not always be easy to scan along the floor to find the wall, IFOM(WALL,FLOOR) will not be as high as IFOM-(FLOOR,WALL). Let us look at the example of a telephone and a table. Given a certain table (say TABLE1), where we know that there is always a telephone, the IFOM(TELEPHONE,TABLE) will reflect the work that may be

required to localize it on TABLE1--this will not be very much, so the IFOM value will be high. If we know further that the telephone is always in a certain location on the table, then IFOM(TELEPHONE,TABLE) will be even higher. However, suppose we know that there is a single telephone on some table in a given room, but that there are three tables in the room. The probability that the telephone is on any given table is only one-third. The value of IFOM(TELEPHONE,TABLE), reflecting this probability, will thus be less than IFOM(TELEPHONE,TABLE1).

The DFOM indicates the advisability of a direct visual search for a feature. In Section III-B we summarized the most important considerations entailed in selecting acquisition attributes. The first consideration was criteriality, the likelihood of the feature being present given that the object itself is. If telephones are always black, then black is a criterial feature; but, if telephones come in several colors, then color becomes less criterial. The second factor concerned the uniqueness of the feature in the world; we prefer features that do not also characterize many other objects likely to be present.

The DFOM must reflect the degree to which prospective features satisfy these two prerequisites. It must also reflect the cost per sample of testing for the feature and the expected number of samples to be examined. The second factor is primarily a function of the size of a feature and the degree of localization within the scene imposed by prior knowledge. We will use whatever spatial constraints we have on the object to "window" the picture (i.e., to limit the area that must be searched for the feature). We will then compute the expected size of the object in that window. The important criteria here will be the size of the feature in three-dimensional space and its expected distance and orientation with respect to the sensor. The ratio of the expected size of the object to the size of the window determines both the expected number of points to be tested and the required sampling density.

The cost of testing each sample depends on the required detection reliability. Reliability will be estimated using simplified detector models that take into account common error conditions affecting the cost-reliability of particular modules. For instance, it is known that highlights, to be expected on glossy objects, will adversely affect the accuracy of both range and hue measurements. The cost of the cheapest detector with adequate reliability will be used in calculating the DFOM. Special purpose modules, designed to be highly cost-effective in very constrained tasks (e.g., a floor finder program) will be selected when available.

A final consideration in selecting an acquisition feature is the prior availability of relevant information in STWM. If, for example, the color or surface orientation of some samples in the search window are already known as a result of previous goals, then the expected cost of another search is correspondingly reduced. The DFOM for such features should be increased accordingly.

To help coordinate current goals and information already in hand, we will store pointers with each detector module (in the detector library) to all subgoals utilizing them. This record of subgoals dependent on the same module, together with a history of which ones have been executed, will enable the planner to determine whether information potentially relevant to a current subgoal is available. Eventually, the GM may be able to use these pointers to dynamically redirect information not pertinent to the particular subgoal for which the information was gathered.

We will conclude this discussion of FOM computation by reiterating the fact that the IFOM computations are based primarily on spatial constraints, while the DFOM computations are based primarily on direct visual features. However, both rely on some common factors like feature size and the criteriality and uniqueness of a feature conditioned

on the presence of other objects. IFOM is a function of three main things--the type of relation between objects, the probability that the relation holds, and the amount of residual work required to localize the constrained object given the constraining one. The DFOM is also a function of three things--the criteriality and uniqueness of the feature, the relative size of the feature, and the utility of the detector for that feature.

We have by now generated a plan for finding an object based on crude estimates of the cost of acquiring objects using their strongest visual characteristics. We have not taken into account any interactions between goals, or possible alternatives that may be available at a node. These factors, along with more precise estimates of FOM will be the subject of the following section on plan refinement.

c. Plan Refinement

During the refinement stage of planning, we wish to take a closer look at subgoals that appeared promising during the initial phase of the planning. We intend to perform more precise checks that, because of cost and level of detail, were inappropriate in the initial stages. One important refinement is to obtain more precise localization of features in the picture. This localization could require relational chains of inference from other more precisely known objects and may result in changed FOM values for various nodes. These precise spatial bounds will be used again during execution to provide starting points and windows for the detector routines.

So far, our estimated FOM for finding an object has been based solely on anticipated costs of direct and indirect acquisition. These estimates must be refined to take account of the expected cost of corresponding validations. If we have several strong characteristics of an object, then we know that it will be possible to increase our

confidence in that object fairly easily. If, however, the object only has a few characteristics, or only weak ones, then it may be more difficult to validate. Thus, a goal with several good acquisition alternatives is more promising than one with only a single acquisition feature, and should have its FOM increased accordingly.

These considerations provide a crude planning estimate of the ease of validation, estimating the ease of increasing confidence of the desired object, but without concern for distinguishing it from other similar ones. During the validation phase of execution, we will consider which of the remaining easily acquired features distinguish as well as confirm the object. However, this level of detail is not necessary at our present stage of planning.

During the refinement of the plan, we will also consider the fact that several branches of a plan may have subgoals in common (e.g., finding the floor might be useful to finding tables, desks, and the wall). Since we can distribute the cost of this operation over several branches of the plan, we can effectively reduce the cost of the individual subgoals. This would increase the FOM of those branches, and perhaps of the parent nodes of the branches.

Earlier, we mentioned that a detector might produce useful information for subgoals other than the one for which it was activated. We will attempt to anticipate the likelihood of this sharing during planning, in order to improve the FOMs of the related subgoals.

An important constraint on module sharing is that the search spaces of the related subgoals must overlap. A reasonable way to group subgoals satisfying this condition is to lay a coarse grid on the picture and note objects whose search space overlaps each cell. When one of the objects is found, the search space of the other objects should be contracted to avoid overlapping the identified portion of the scene.

Reducing the search space will then cause a corresponding increase in the FOMs of these remaining objects.

We have now concluded the initial stage of planning to find an object. We have, as a result of this, a planning tree for each desired object. At each node of the tree, we have computed a static FOM value to guide us in our choice of execution alternatives. As new information is acquired, we expect that these values will change, and also that the expected execution path will change. We now proceed to a discussion of how this tree is used in the execution of the plan.

4. Execution

We would like to preface our discussion of the execution phase of our system by indicating the principal distinction between our concepts of planning and of execution. In planning, the only information available to the system is that which is already in VM (due either to manual model building or to previous processing) or in the STWM (due to recent execution). During execution, we will also perform "planning", but this will primarily use new information acquired by processing TV or range pictures. That is, during execution we can draw on information gained by the application of detectors to images (which may be new or old).

We will begin our description of the execution phase of the system with a brief overview. Following this overview will be a more detailed description, and finally an example.

a. Overview

The execution will start with the selection of the best (i.e., highest FOM) object to find, from the list of desired objects. From the plan for finding that object, we will select the best object to look for directly. We will then choose a part of that object, and the best feature of that part (the PRIMARY A-FEATURE) for acquisition

in the picture. The image will be examined by sampling coarsely, and looking for a sample with the appropriate characteristics. When one is found, the system will attempt to grow the part using the PRIMARY A-FEATURE and any other useful features of the part.

After acquiring the part, we will begin to validate by performing some immediate tests to increase our confidence in the fact that we have found the desired part. Validation will then proceed by attempting to distinguish the desired object from other objects having a similar part.

When we have sufficient confidence in the object which we have found, we will proceed back up the planning tree to attempt to satisfy the goal that had this object as a subgoal. When we have either satisfied all assigned tasks, or exhausted the budget, we terminate.

During the course of execution, new information will cause conditional adjustments in our FOM estimates. We now describe how these adjustments may affect the flow of control. The GM will examine information obtained in the course of execution, decide which goals are affected, and compute appropriate adjustments to their FOMs. Whenever the FOM of a subgoal is modified, the new value is backed up the planning tree, making FOM adjustments at each level, until a top-level goal is reached, or until no further adjustment is necessary.

FOM modifications will usually be backed up only when we terminate the current subgoal (either successfully or unsuccessfully). This will eliminate a tendency to flit from goal to goal, and will thus minimize expensive overhead functions.

At this point, based on the updated FOMs, the planning budget must be reallocated and a new subgoal selected. We expect that when a subgoal succeeds, the goal that initiated it will usually be the most promising thing to do next. However, if another subgoal should have a significantly better FOM, it would be selected for attention. If

no goal appears sufficiently promising (i.e., the FOMs are all too low), the GM may decide that replanning is the best alternative; it can recall the planner, either to awaken suspended nodes or to reevaluate existing nodes in a new context.

b. Acquisition

The first step in execution will be the distribution of budget among the set of top-level goals similar to the resource allocation for planning. In this case, however, budget will be allocated on the basis of both I values and FOM values for the goals. Goals will be allocated a percentage of the budget proportional to the ratio of the product of their FOM and interest, normalized by the sum of such products over all the top level goals. The goal with the highest allocation will then be selected for execution. Execution will proceed by descending into the planning tree and, at each branch, selecting the subgoal with the highest FOM value. Budget passed down from the previous level can be distributed among the subgoals, at each node, in a number of ways; for instance, in proportion to their normalized FOM values. However, in the initial implementation we will simply pass on the total remaining allotment to the best subgoal at each node. As FOM values change during execution, remaining budget will be dynamically reallocated to the currently best subgoal.

This initial phase of execution proceeds down the tree until a node is reached that represents an object that can be looked for directly. The best feature of the most promising part of the object will be chosen to be the PRIMARY A-FEATURE. The rest of the features will be SECONDARY A-FEATURES, or VALIDATION FEATURES.

The scanner will be called with the PRIMARY A-FEATURE, and the "starting information" for the part. This starting information will consist of the window within the picture where we have localized

the part, and information about the scan, such as the order and density of sample points.

The scanner will select the coordinates of the next sample point in the picture, and then test the local sample characteristics for correspondence with the PRIMARY A-FEATURE. For example, we might be looking for a table, using its horizontal surface as the PRIMARY A-FEATURE, and its color as a SECONDARY A-FEATURE. The starting information may cause us to examine only the lower half of the picture, and to sample it with a 3 by 3 operator applied at every 15th picture-point. A typical (default) sampling raster will be from left to right across the window and then bottom to top. A 3 by 3 surface will be fit at each sampled location to determine whether a horizontal surface of the appropriate height might be present.

If the scanner is unable to find any samples with the appropriate characterizations, it exits with failure. The GM will then cause failure of all nodes in the tree that required that feature. This, in turn, might require the FOM values of all the nodes in the tree above these pruned nodes to be updated. At this point, the executive will choose whether to use a SECONDARY A-FEATURE as an alternate acquisition test, or to pursue another goal. The choice depends on the revised FOM values.

If the SCANNER succeeds in finding a suitable surface characterization, the successful sample will be checked to see that it has the appropriate SECONDARY A-FEATURES. In the case of a table, if a sample had the right surface height and orientation, the next step would be to check the color.

If the SECONDARY A-FEATURES all match, then the global characteristics of the part must be checked. If any of the secondary features (or subsequent tests) fail, the GM will still make note (in a

"sample table") of those characterizations that do succeed. Later, the FOMs of other subgoals characterized by the successful features can be increased.

Now, assuming that a good candidate for the desired part is found, the scanner will be suspended (so that it can be restarted later) and the global attributes of the part extracted. We will base our discussion on surface parts that appear as "regions." Parts need not be surfaces--to locate the wall it may be wise to search for the wall-baseboard boundary--however, the remarks below will still apply. In general, the system will need to use a priori information to successfully extract a part. Thus, it might seek to extract a region with specific color, surface orientation, and textural attributes. A detailed discussion of goal directed region extraction follows in Section III-D.

c. Validation

The initial step in validation is to compare the global attributes of the extracted region with those of the desired part. Some of the region properties that can be compared are size, shape, and uniformity of color. This initial validation step will establish a basic level of confidence, indicating that what was found at least looks like the desired part.

If any of these global tests fail, several options are open, depending on the current level of confidence in the part. First, if the overall confidence is low, the validation should fail. In this case, control returns to the scanner, which is restarted from its suspended state. The GM first will update FOM values of goals which are affected by the information obtained during the suspension.

The second option for marginal confidence is to examine the reasons for failure of the test. Thus, the system might hypothesize

and check for occluding objects. If the hypotheses prove to be true (or at least plausible), then validation could continue with other features. Alternatively, it may be appropriate to vary the criteria used in region extraction and try regrowing it. We would then reenter the validation stage with the new region.

The final choice is simply to ignore the failure and continue validating other subgoals. We would, of course, select this option if the confidence were already high enough to accept the region as being the required part. However, it also seems reasonable to suspend, at least temporarily, any error diagnosis whenever there are other reasonable alternatives to pursue. The need for diagnosis effectively makes the original subgoal very expensive. Therefore, it should be abandoned until all the more promising alternatives have been exhausted.

Now that the extracted region is known to possess the visual attributes of the described part, it is necessary to ascertain whether that part, in fact, belongs to the desired object. To accomplish this, we will compare the object with other objects in VM having similar appearing parts. (These objects make up the "ambiguity set" of the desired object.) A set of object attributes will be selected, which distinguish the desired object from others in the ambiguity set. These attributes will include parts and features which the desired object must have, and those which it specifically must not have.

Even if the ambiguity set is empty, there is still the possibility of accepting an unknown object that happens to have some visual attributes of the desired object. To minimize the chance of such a mistake, it is desirable to check other parts of the desired object. Additional parts of the object, beyond those necessary to confirm recognition, may also be needed to locate boundaries more precisely.

The relevant criteria for deciding which feature to pursue in validation happen to be the same as those for selection of acquisition attributes--i.e., a feature that is criterial, distinguishing, and easy to obtain. Therefore, both the selection and the search for validation features can be accomplished by recursively applying FIND to the remaining features of the desired object, in the context established by those already validated. Features will again be selected in an order determined by the product of their estimated FOMs and I values. Here, I values reflect the extent to which each validation goal raises confidence in the object.

The limited context simplifies these subordinate finds in two important ways. First, the area that must be searched is restricted to the immediate vicinity of the previously acquired parts. Second, the selected feature need only distinguish the desired object from remaining members of the ambiguity set.

Validation of a tree branch terminates when either the alloted budget is exhausted, the required confidence in the goal is achieved, or the confidence drops below a failure level. If validation succeeds, control passes to the next most promising (and adequately funded) subgoal. If it fails, control returns to the scanner, after the GM has updated affected FOM values.

d. Example

We would like to conclude this discussion with an example illustrating a possible flow of execution. The goal will be: Find a telephone. Let us assume that during planning the tree shown in Figure 11 was generated. In this tree, the FOM values--chosen arbitrarily for purposes of illustration--are written next to the nodes, and the IFOM values--also illustrative, are written beside the branches. Note that in several cases the FOM value shown for a node is greater than what

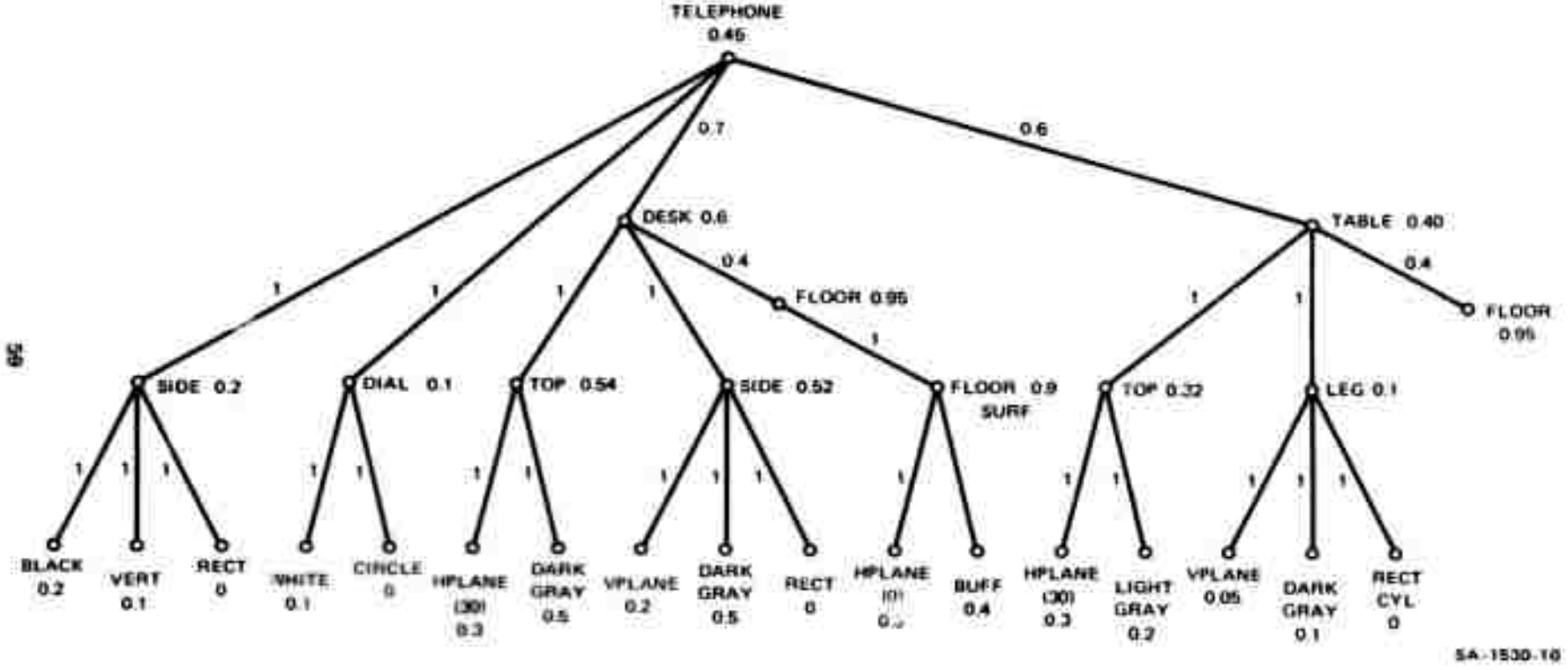


FIGURE 11 PLANNING TREE FOR FINDING TELEPHONE

would be obtained from backing up the maximum descendant FOM. Such an increment would have been added during plan refinement to reward nodes with several good alternative subgoals (e.g., the desk can be found by looking for its top or side) or those that are subgoals to several other nodes (e.g., the FOM of the floor is increased due to the fact that it appears twice). During the development of this example, the quantitative adjustments of FOMs will be somewhat arbitrary; what we consider important here is the qualitative effect of these adjustments. The discussion will not include budget allocation; assume that sufficient budget has been allotted to perform the task.

Now, proceeding down our tree, we see that the best way to locate the telephone is to first find a desk. In looking for a desk, the top is the best part for acquisition, and the color, dark gray, is the most promising feature. Therefore, we start the scanner (with any available starting information) in an attempt to find a dark gray sample. Let us assume that it fails. The GM now changes the FOMs of nodes requiring a dark gray region to zero and prunes those branches. After the adjustment of all affected FOMs, we have the tree shown in Figure 12.

The desk is still the most promising way to find a telephone, but this time the floor should be used as an aid to finding the desk. The scanner thus attempts to locate samples belonging to a horizontal plane of height 0. Assume it succeeds both in finding such a sample, and in growing a surrounding region based on these surface attributes and the color (buff). The GM now readjusts the FOMs of affected subgoals to reflect success in finding the floor.

In the new tree, shown in Figure 13, the desk remains the best way to proceed to our goal of a telephone. We should point out that this is because the side of a desk is more distinctive than the leg of a table. The scanner now attempts to locate a horizontal plane 30 inches above the floor, in search of the desk top. Assume that a sample of such

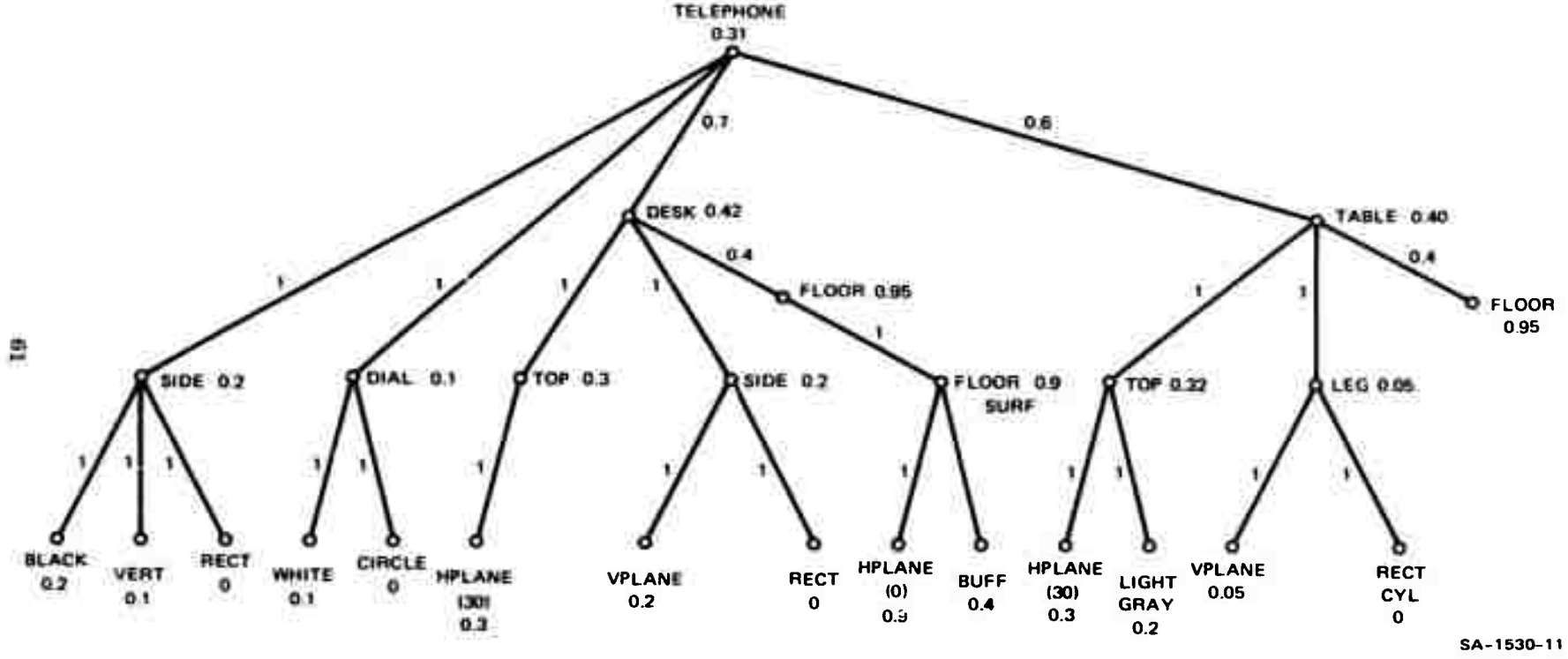


FIGURE 12 TELEPHONE PLANNING TREE AFTER FAILURE OF DARK GRAY COLOR DETECTOR

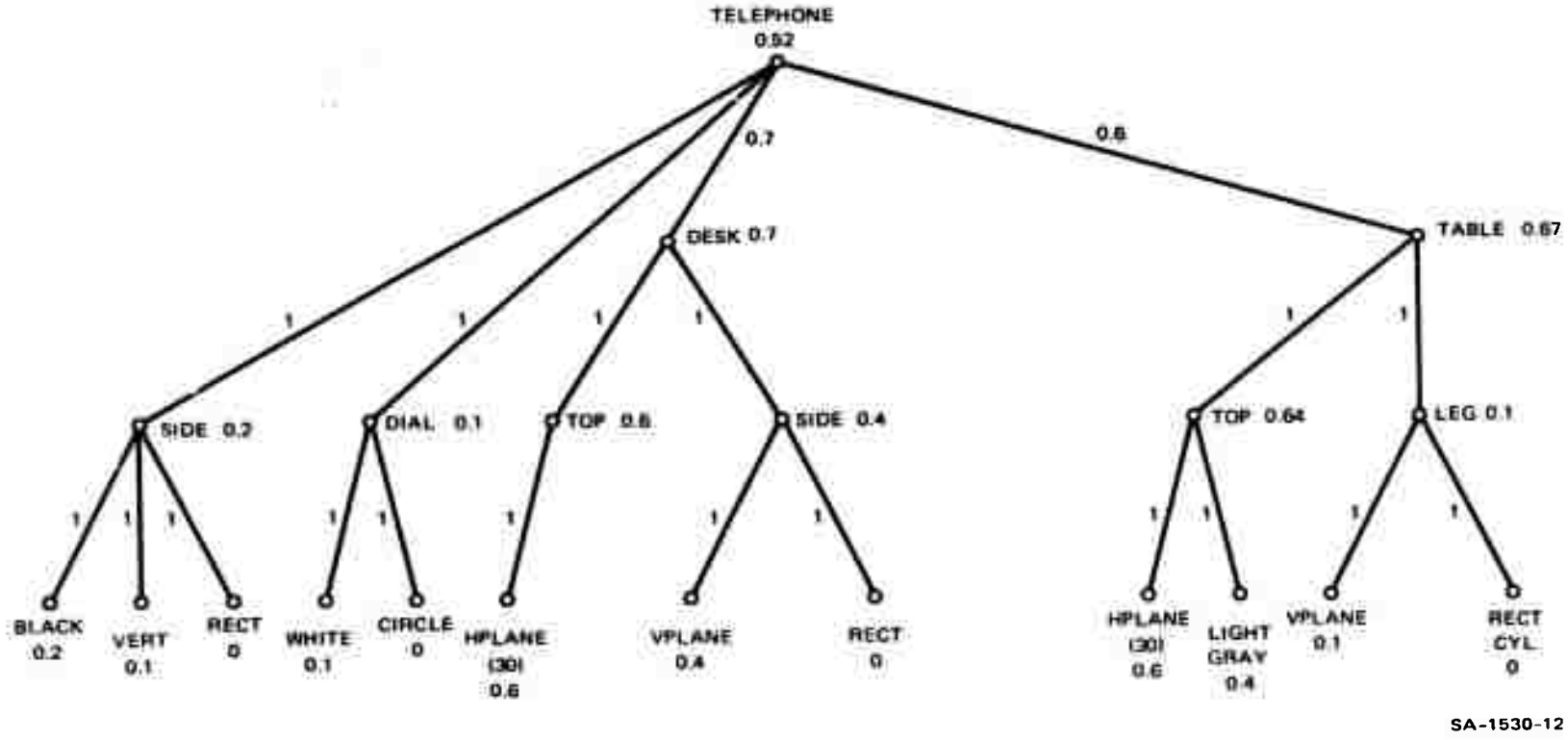


FIGURE 13 TREE AFTER FINDING FLOOR

a plane is found, but with the wrong color (even though color was removed as an acquisition feature, it is still a requirement for validation). This fact is noted in the sample table, and the scanner continues. It ultimately terminates, never having found a horizontal plane of the appropriate color. However, the several samples of a horizontal plane at a height of 30 inches that were found will increase the FOM of the HPLANE(30) feature of the top of the table. The resulting planning tree is shown in Figure 14. The top of the desk has been abandoned as an acquisition feature, but the side still exists as a subgoal. If the side is ever located, then the characteristics of the top can still be used in validation (with more tolerant acceptance thresholds).

The top of the table is now the strongest subgoal. The sample table provides the location of the samples of horizontal plane that were observed, and a check indicates that the color matches that required for the top of the table--light gray. The complete region is grown, and its size and shape are successfully compared with the expected dimensions of the top. The validation of the table is completed by finding its legs. Successful validation of the table causes another flurry of FOM reevaluation, resulting in the new tree of Figure 15.

It is now cost-effective to look directly for the telephone within the boundaries of the table top, using the "black side" as the best feature. This color is located, and it is verified that the sample belongs to a vertical surface. The region is then grown and found to have appropriate shape. Finally the dial is sought to complete validation and pin down the location of the telephone. (Of course, by now our caller has hung up!)

This example shows how the output from the planning stage was processed during execution. The GM performed substantially the same evaluations that were required during planning. While there was no actual reincarnation of the planner at a suspended node, it would probably have

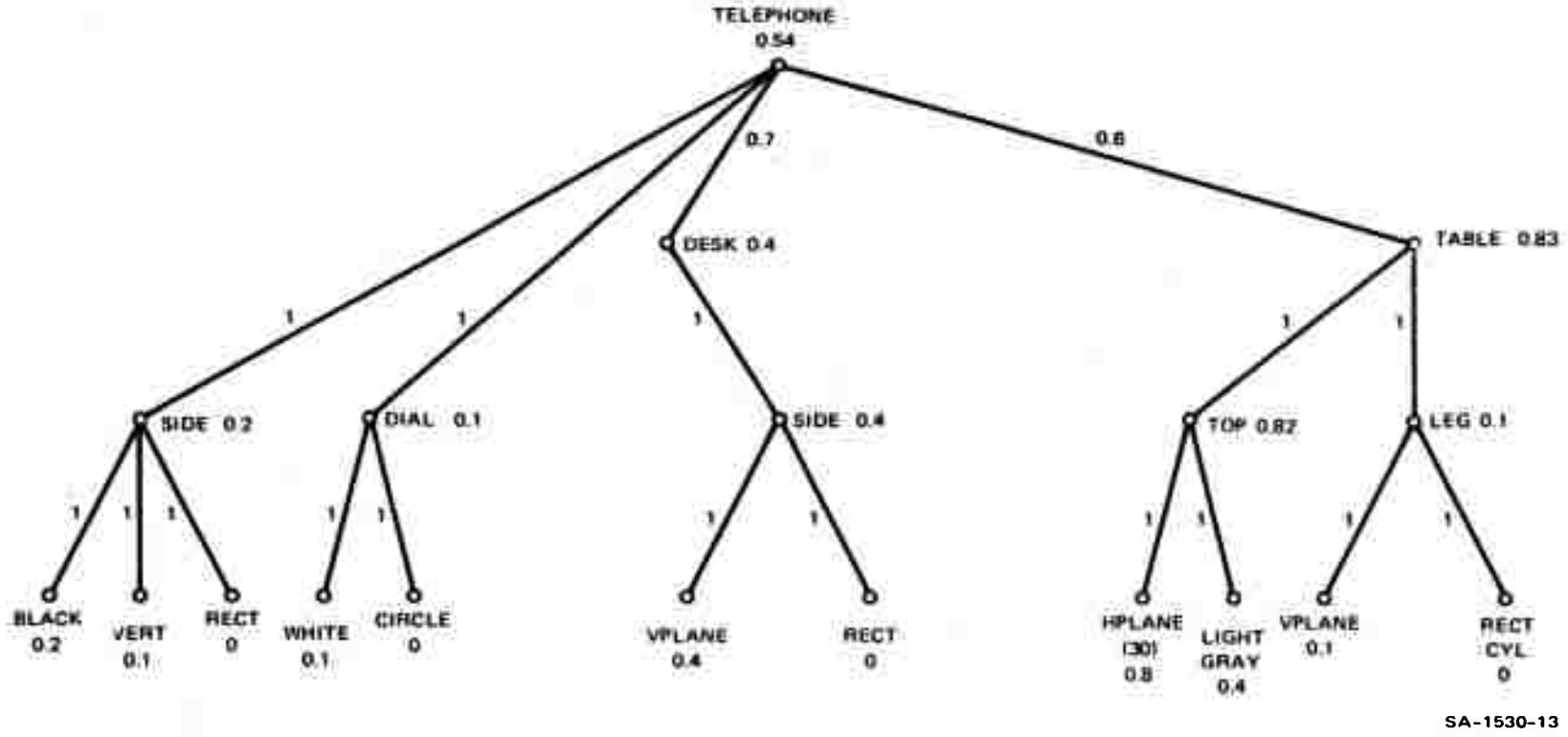


FIGURE 14 TREE RESULTING FROM LOCATION OF HORIZONTAL PLANE

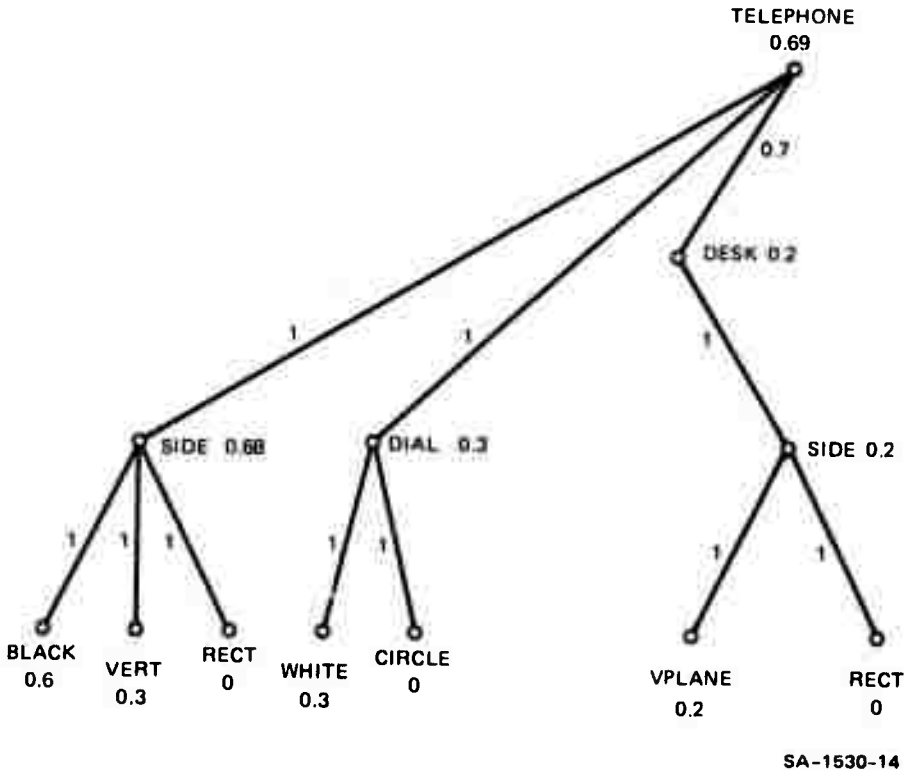


FIGURE 15 TREE AFTER FINDING TABLE

been necessary to resort to this if the table had not been found. At each stage, the executive selected the best goal and the best acquisition features to pursue. We expect to implement a system that functions essentially in this way, in the coming year.

D. Goal Directed Scene Segmentation

One of the most interesting aspects of the strategy outlined above is the degree to which high level recognition and low level feature extraction have been integrated; one knowledge base and one basic strategy concept are used at all levels of the perceptual process, from the selection of attributes for region growth to the selection of distinguishing features for object recognition.

In this section, we want to elaborate on our goal directed approach to scene segmentation (i.e., region extraction) based on multisensory

data. The success of any perceptual system is largely dependent on the accuracy and appropriateness of the visual features it extracts. We believe we have found a straightforward way of utilizing high level knowledge to improve substantially the reliability with which regions can be extracted.

1. Background

"Segmentation" is the process of partitioning a scene into interesting figures distinguished from the background. This process is essential to virtually every perceptual function. Robot projects have hitherto largely avoided the problems of segmentation by choosing textureless environments in which both objects and background could be adequately characterized by regions of homogeneous brightness. A typical algorithm for segmenting such an environment was:

- Group contiguous raster points into regions of identical brightness.
- Merge adjacent regions if the contrast over a suitable fraction of their common boundary is sufficiently low.
- Merge recursively until no more combinations are possible.

Such an algorithm cannot cope with intensity variations due to color, texture, and range that characterize most real-world scenes. A superficial solution is simply to incorporate additional primitives (e.g., texture operators) and/or sensory data (e.g., color and range) into the initial similarity judgements. However, this solution exposes serious limitations that are deeply rooted in the bottom-up nature of the strategy.

In a bottom-up approach, one would lump the new primitives and sensory data together into a weighted attribute vector and judge similarity by distance in a multidimensional space. However, in any particular context, some of these attributes will be irrelevant, causing

unnecessary fragmentation of surfaces that are actually homogeneous with respect to an appropriate attribute set.

This fragmentation, in addition to the fragmentation caused by ambiguities inherent in local similarity judgements, must again be resolved by merging similar regions at a subsequent level of processing. However, in naturally textured scenes, it is necessary to consider combining proximate as well as contiguous regions, according to criteria that may include size, shape, spatial orientation, spatial relationships with other regions, and so on, in addition to the original attribute vector. A lumped computation here will obviously include inappropriate factors. An alternative is to try various combinations of attributes and see which provides the "best organization" according to some Gestalt type criteria. Unfortunately, this alternative is infeasible combinatorially with a serial computer.

2. Classification Approach

Scene segmentation at all levels must be responsive to higher level intent and expectations. In our goal directed vision system, regions are grown hierarchically by selecting more primitive regions satisfying specific criteria associated with the object of interest. The lowest level regions are grown by selecting proximate image samples having local attributes (i.e., color, surface orientation) characteristic of a particular object.

Growth by selection brings high level knowledge to bear on the problems of low level feature extraction in an intuitively nice way. Regions can be grown on the basis of attributes that both unify the desired object and distinguish it from surrounding surfaces. Our basic strategy for choosing simple distinguishing features can be directly applied to choosing selection criteria for scene segmentation. In fact, the same criterion originally used for acquisition will often be

appropriate as a selection criterion for extracting the surrounding surface. This criterion may be improved, however, by using knowledge of surfaces likely to border physically on the desired surface (e.g., door is adjacent to wall and floor, door knob is surrounded by door).

When specific surface adjacencies are known, the selection decision can be based on a direct discrimination of their distinguishing attributes. A sample is accepted if its relative feature-space distance is closer to the characteristics of the desired object than to those of the known alternatives. Such discrimination tests tend to be much more tolerant and reliable than one-sided acceptance thresholds.

The optimal discrimination criteria will usually be directionally dependent. For example, when extending a door region downward, it may be best to ascertain whether the local orientation of the next sample is more nearly vertical or horizontal (the latter indicating a floor). However, sideways expansion might have to be based on a brown versus buff color discrimination, since the wall is also vertical, and probably at an unknown angle to the door.

Surface adjacencies can be determined empirically when a region, in the process of growth, intersects a previously identified surface. The common border, which was originally obtained by criteria selected independently for each surface, can then be refined by reclassifying samples in the vicinity according to a discrimination criteria based on both surfaces.

Many other characteristics of the desired object can be brought to bear as additional constraints governing sample selection. For example, the area from which samples should be selected is bounded by the expected region size, while texture scale specifies the required sample density. The acceptance thresholds for selection can be set from knowledge of the expected uniformity of an object's surface attributes and of expected

distortions introduced by the associated detection processes. Since the resulting thresholds are absolute, selection is not subject to the ambiguity of local similarity judgements.

3. Coping with Texture

Textured surfaces provide the clearest illustration of the advantages of a classification approach to scene organization. It is simply not feasible to extract arbitrarily textured surfaces, bottom-up, on a serial computer because of the impossibly large number of organizational hypotheses that would have to be tested to discover the criterial pattern. Consider, for example, the effort that might be required to determine, bottom-up, that the floor region should not be segmented according to the shape of linoleum streaks.

The combinatorial problems associated with serially discovering two-dimensional patterns can be overcome by looking top-down for the known criterial attributes of a desired surface. Raster points can be selected in groups, satisfying explicit statistical or descriptive criteria, such as a required distribution of colors within a specified radius, or even a detailed prototype micropattern with specified spatial relationships among the colors.

Many textures admit to a hierarchical organization, consisting of primitive regions that are grouped into subpatterns, which in turn are grouped into patterns, and so on. For these surfaces, the selection process can be applied recursively. Thus, a checkered texture might be organized by first selecting contiguous samples of specified colors to get the primitive regions (i.e., the squares) and then selecting the set of resulting regions that are contiguous, are of appropriate size and shape, and alternate in color.

This approach is applicable to organizing several objects in our scenario. For example, the floor can be grown by collecting sets of contiguous brown or black samples (corresponding to linoleum streaks) and then collecting the set of such regions with appropriate size (shape arbitrary) that are surrounded by buff. Similarly, a picture could be extracted within the context of a vertical wall plane by selecting a contiguous set of nonbuff regions and small buff regions completely surrounded by non-buff ones.

Conceptually, the selection process can be recursively applied through many levels of hierarchy to extract arbitrarily complex patterns. However, in limited contexts, a single level organization based on simple distinguishing features of the desired texture may prove more effective. For example, a highly textured object on a plain surface might be best extracted as a region of many small regions; the detailed structure of the texture is irrelevant here, and can be ignored. In another environment, the same object might be most easily extracted by selecting samples on the basis of their proximity to a distinguishing color that happened to be a component of the texture.

The above representations are admittedly ad hoc and of limited generality, but they provide optimum knowledge-based partitioning in those environments. Region growth, like recognition, can often be accomplished more cheaply and reliably using simple discriminations between context limited alternatives than by exhaustive description.

4. Operational Details

Most surfaces can be extracted on the basis of several independent criteria, such as color, texture, or orientation. In practice, we shall choose the simplest logically sufficient set of attributes to grow the initial region, saving the remaining alternatives for error recovery. The selection criteria for each attribute can be combined into

a weighted vector or logic function which is used to test each sample point. A logic function seems preferable because absolute constraints like "must" and "must not" can be applied to specific attributes and because the problem of scaling unrelated criteria is bypassed.

Regions will initially be grown with coarse resolution, relying on crude measures of size and shape to eliminate cheaply many bad hypotheses. Detailed boundaries can then be obtained, if necessary, by using the attributes of the crude surfaces as classification criteria. Additional image samples obtained at finer resolution are simply assigned to whatever neighboring surface they most closely resemble.

5. Error Recovery

Although our classification approach can be expected to extract regions more reliably than past bottom-up techniques, we forsee the continuing necessity of an iterative growth process, guided by feedback from unsuccessful validation tests.

a. Background

Past attempts to graft such feedback onto bottom-up systems have largely failed because knowledge of an error was often available only after a global recognition decision failed. This was too far removed from the source of error to pin-point the cause explicitly. It was also difficult to model conventional region growing algorithms so as to establish the relations between detector idiosyncracies, known object attributes, region-grower parameters, and the resulting regions. Consequently, it was difficult both to diagnose the cause of suspected errors and to alter parameters given an assumed cause.

By contrast, our goal directed perceptual strategy attempts to validate the expected attributes of a region (e.g., size, shape,

uniformity) as soon as it is grown. For example, evaluation will be performed on regions extracted at each level of a hierarchically organized texture. Suspected errors can be directly related to specific selection criteria, based on world knowledge and pragmatic experience with the various sensory processes. Feedback from this diagnosis is then treated as simply another goal directed factor influencing the refined selection criteria.

b. Diagnostics

The type of knowledge-based diagnostic process we envisage is illustrated by the following list of plausible explanations and corresponding recovery strategies for a region that is smaller than expected:

(1) Occlusion--This hypothesis is easily established given range information; occlusion is likely when the visual boundary of a selected region is bordered by samples of nearer range. Additional samples that satisfy the classification criteria and are within the expected size boundary, but are not contiguous to the originally selected set (due to occlusion), can then be included in a refined size estimate with reasonable confidence.

(2) Holes--Explanations for holes in a supposedly homogeneous region, not explained by occlusion, can be sought by checking whether any attributes of the expected region excite known limitations in the sensory domains used for selection. For example, color data can be anticipated to be unreliable for pastel (weakly saturated) objects, and at highlights expected on glossy objects. Range data will also be unreliable at highlights as well as on dark or very oblique surfaces (because of inadequate reflected light). Such explanations are easily checked and can often be rectified by looking for continuity of an

alternative attribute not involving the affected sensor.

(3) Gradients--Illumination and texture gradients frequently distort homogeneous surfaces due to artifacts inherent in the imaging process. As a result, static acceptance thresholds often fail. Suspected gradients can be extrapolated to correct the selection threshold as a function of position. One can also obtain the surface orientation with range data and use it with an appropriate imaging or illumination model to compensate for the gradient analytically. Brice and Fennema[6] suspected intensity gradients whenever the boundary of a region was weak (i.e., low contrast) and wiggly, a so-called quantization contour. Contiguous regions with such boundaries were merged since, individually, they did not correspond to known objects. Although this heuristic was originally intended for use in a bottom-up algorithm, the characteristics "weak" and "wiggly" can be used by our system like any other region grouping criteria.

A more general heuristic is based on the fact that quantization contours are unlikely to affect two independent sensory modalities at the same points on an object. Thus, continuity in another attribute (e.g., surface orientation) can justify ignoring weak boundaries among the original (e.g., color) regions.

(4) Wrong Object--The most obvious cause of error is simply that an undesired surface was acquired. The global attributes of the extracted region can be checked to determine if they correspond to any surface of an actual known object. If so, the surface should be regrown based on local surface attributes that distinguish this other object from the desired one. Where no such attributes exist, discrimination must be accomplished in terms of other parts of the objects, by a higher level validation strategy.

These error hypotheses are only representative. Similar error recovery lists have been compiled for regions that are too large or of the wrong shape.

c. Diagnostic Strategy

Diagnosis will be performed by systematically investigating possible errors. The order of examination will be empirically based on the frequency with which each error is observed in the current environmental context. We have yet to decide whether statistics will be kept only for a general class of errors (e.g., wrong object) or for very common specific errors (e.g., floor often merged with wall due to similar color).

It may seldom pay actually to pursue such detailed diagnoses. Diagnosis is most important in cases where high confidence requirements demand a positive explanation for errors or where exact surface boundaries are needed (e.g., for manipulation). Otherwise, it should be more effective simply to exercise available strategy options without regard to the cause of original failure. For example, the system could attempt to regrow the region using other attributes and/or sensory modalities. Alternatively, the initial failure may make it more desirable to attempt validation in terms of other surfaces. In either case, subsequent successes or failures should quickly establish a definitive confidence level. Tests for the most common error conditions (e.g., for occlusion or inconsistency of regions extracted with different attributes) could be built into the initial growing process to increase its basic reliability.

E. Multisensory Data

An important goal of our perception research is to make effective use of multisensory data. The initial system is designed to work with

coordinated arrays of range values and intensity values seen through a number of color filters. In this section we summarize our progress in interpreting range and color data. We also describe a technique for interactively generating coordinated arrays of color and range data which we will use to obtain experimental data until our new rangefinder, to be described subsequently, is available.

1. Analysis of Range Data

Range data should ultimately allow simple solutions to some of the most confounding problems now faced by passive vision systems. We are currently experimenting with algorithms to obtain the position and orientation of a surface element for use in acquisition and goal directed region extraction as described above. As a step toward extracting additional useful information, we have analyzed some general problems concerning the low level interpretation of range data as planar surfaces. We expect this analysis to lead to special purpose routines for rapidly extracting certain common types of surfaces (e.g., a horizontal surface of known height). Moreover, the analysis may provide improved algorithms for bottom-up scene segmentation. A reliable, a priori knowledge of a principle planar surface in a scene (e.g., the floor) could help the planning algorithms immensely in tasks requiring some description, especially when an object has to be found in an unknown environment.

The analysis is divided into five topics: (1) alternative methods for scanning and measuring quantized range data; (2) extraction of image points belonging to horizontal and vertical surfaces; (3) classification of planar-region image boundaries and methods for finding points along those boundaries; (4) adverse effects of errors in the range data on the resulting image boundary points; and (5) line-fitting methods for determining the vertical, horizontal, and three-dimensional boundaries in the scene. The analysis, whose details are described elsewhere,[7] is summarized as follows.

a. Range Data

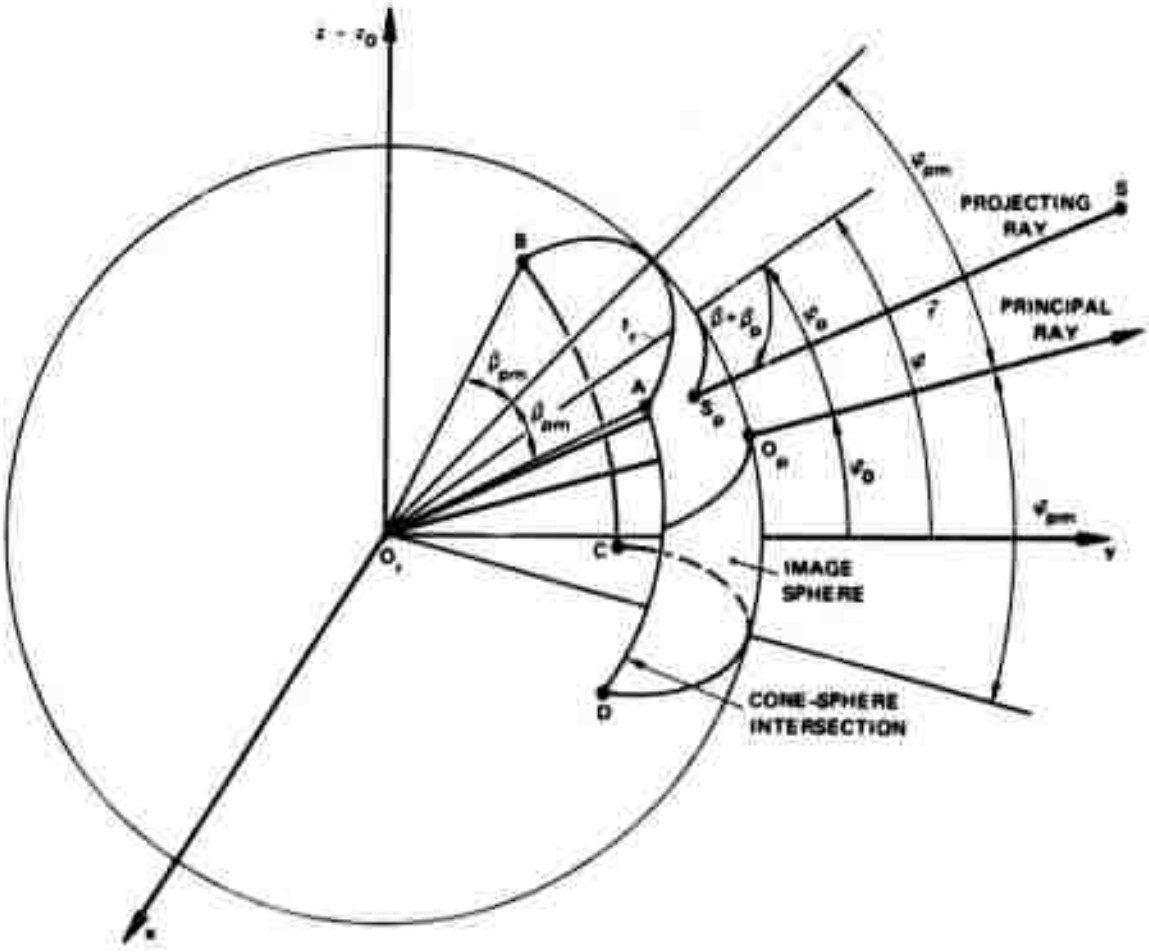
Referring to Figures 16(a) and 16(b), two types of image coordinates, centered at the range finder, are distinguished: spherical, r,β_p and φ_p, whose angular increments $\Delta\beta_p$ and $\Delta\varphi_p$ are constant, and planar, r,x_p and z_p, whose increments $\Delta\beta_p$ and $\Delta\varphi_p$ are tangent adjusted so that the increments Δz_p and, for given z_p, Δx_p are constant. The range data are confined to a solid angle defined by ($|\beta_p| \leq \beta_{pm}$; $|\varphi_p| \leq \varphi_{pm}$).

There are two types of optical range finder, each consisting of a transmitter and a receiver of a light beam, to consider: one is based on the trigonometry of transmitter-receiver-object triangulation; the other is based on splitting a transmitted amplitude modulated light beam into two beams and measuring the phase shift between one of them and the scattering of the other from the object. (Both types of range finders are discussed more fully in Section V.) For a triangulation range finder, missing data (called "shadow gaps") occur for some image points (I,J) if the corresponding object points are seen by the receiver, but are not illuminated by the transmitter (I and J are the indices of the array column and row, respectively).

Six types of errors in the measured range data, $r^*(I,J)$, are recognized: inherent inaccuracy of the range finder, shadow gaps, data dropout, quantization error, noise, and gross sporadic error.

b. Image Points of Horizontal and Vertical Surfaces

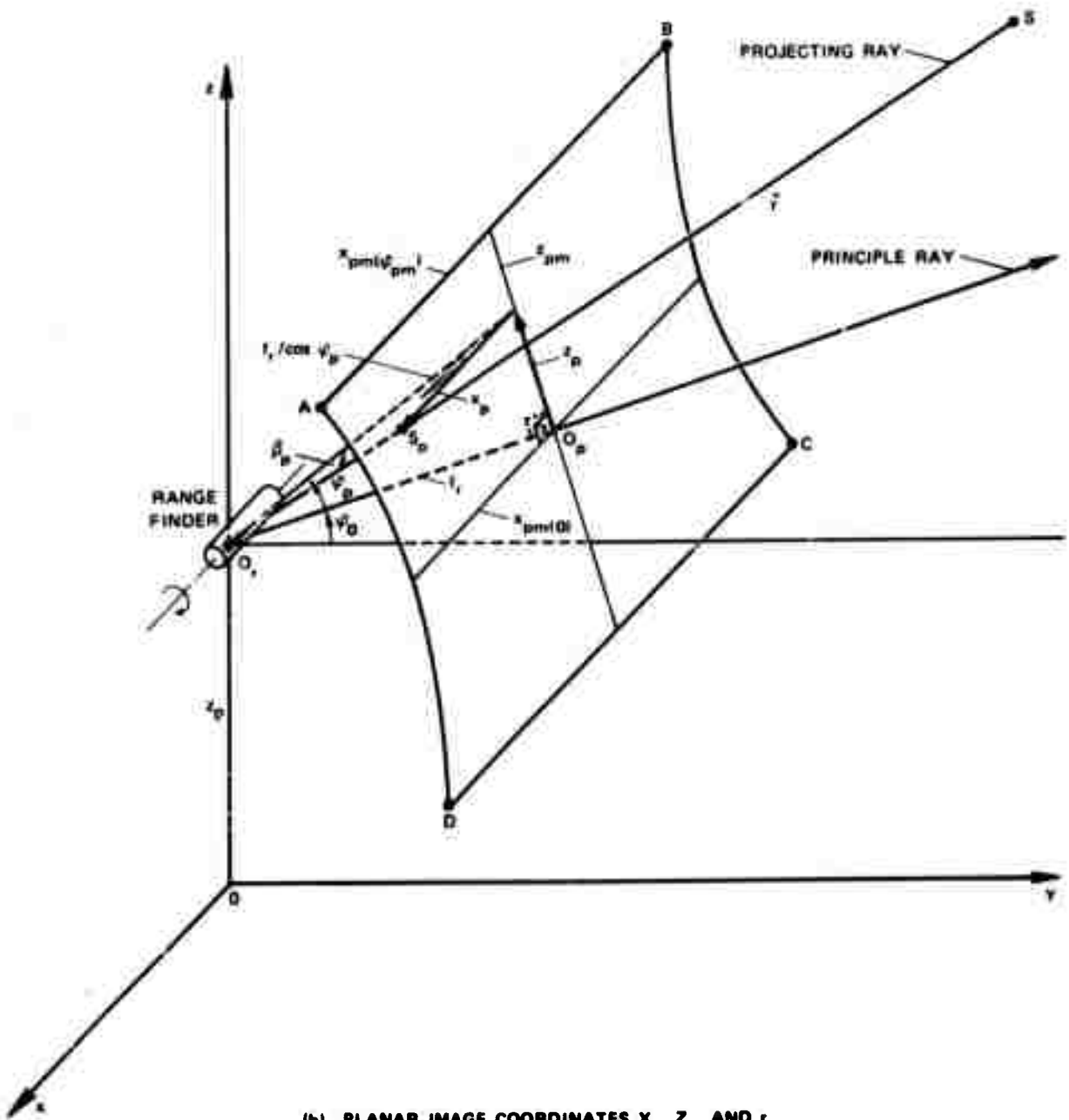
A general plane equation, using range-centered image coordinates, has been derived and then simplified for horizontal surfaces and the floor, and for vertical surfaces. Consequently, a method is proposed for fast extraction of those image points belonging to either the floor or a horizontal surface of height $z_H \pm \Delta z_H$. The method is based on the observation that, for these surfaces, the function $r(\beta_p)$



(a) SPHERICAL IMAGE COORDINATES β_p, φ_p, AND r

SA-1530-1a

FIGURE 16 RANGE-FINDER CENTERED IMAGE COORDINATES



(b) PLANAR IMAGE COORDINATES X_p, Z_p, AND r

SA-1630-1b

FIGURE 16 RANGE-FINDER CENTERED IMAGE COORDINATES (Concluded)

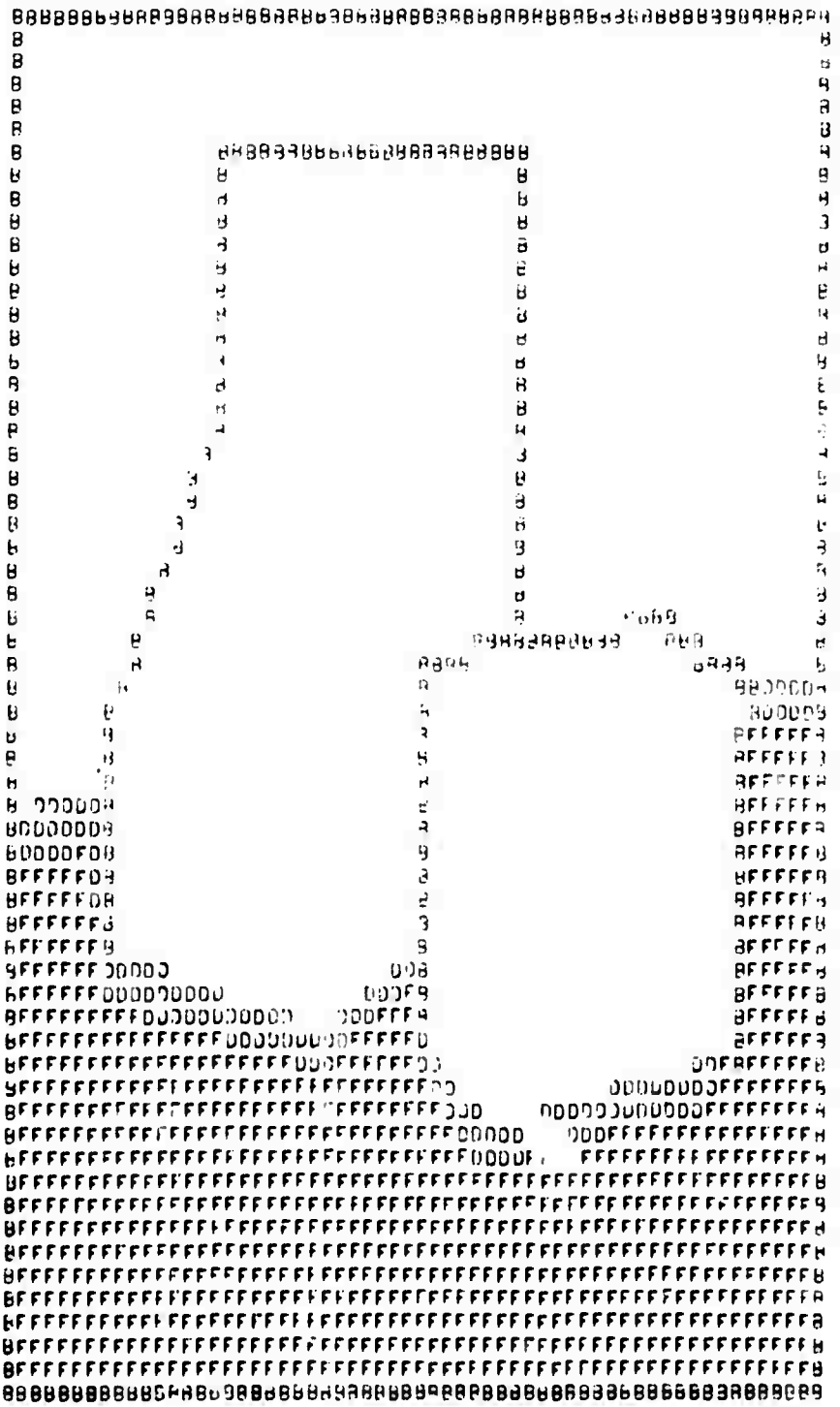
for a given φ_p is symmetrical about its minimum value, r_{min}, at $\beta_p = 0$ and is so flat in the range $-\beta_{pm} \leq \beta_p \leq \beta_{pm}$ that it may be approximated by a constant. A floor point is detected if $r^*(\beta_p) \geq \alpha\, r_{min}$ where $\alpha \leq 1$ is a safety factor. This algorithm was applied to ideal range data (see Section III-E-3) of a scene consisting of a floor, a box, a polyhedral object, and a wall. Using $\alpha = 0.98$, the results are shown in Figure 17. Computed floor points that agree with the true ones are marked by "F," and those computed points that differ from the true ones are marked by "D." Cells marked by "B" indicate surface boundary points, to be discussed in the following section.

Referring to Figure 18, a point on a horizontal surface below the horizon is detected if $r^*(\beta_p)$ is between $r = r^{min}$, corresponding to $z_H + \Delta z_H$ and $\beta_p = 0$, and $r = r^{max}$, corresponding to $z_H - \Delta z_H$ and $\beta_p = \beta_{pm}$. Additional tests are employed in order to eliminate wrong points that pass this test. A method is also proposed for detecting horizontal surfaces of unknown heights, finding their heights, and extracting their image points. The method is based on computing z for each data point, quantizing the z values into a one-dimensional array of cells, and detecting cells with high counts. To prevent missing a high count due to quantization, this process is performed twice, using two arrays that are offset by half a cell relative to each other.

The above method can be extended to detection of vertical planes. Here, the x and y coordinates of all the data points are first computed. High density clusters of (x,y) points are then fitted into straight segments, using a window-count method or a modified Hough transformation method.

c. Image Points of Different Boundary Types

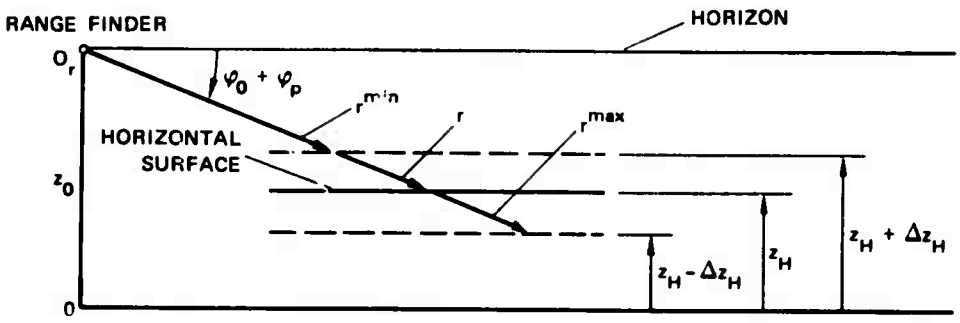
Five types of region boundaries are distinguished: frame boundary (of the range data "picture"), jump boundary (where one object



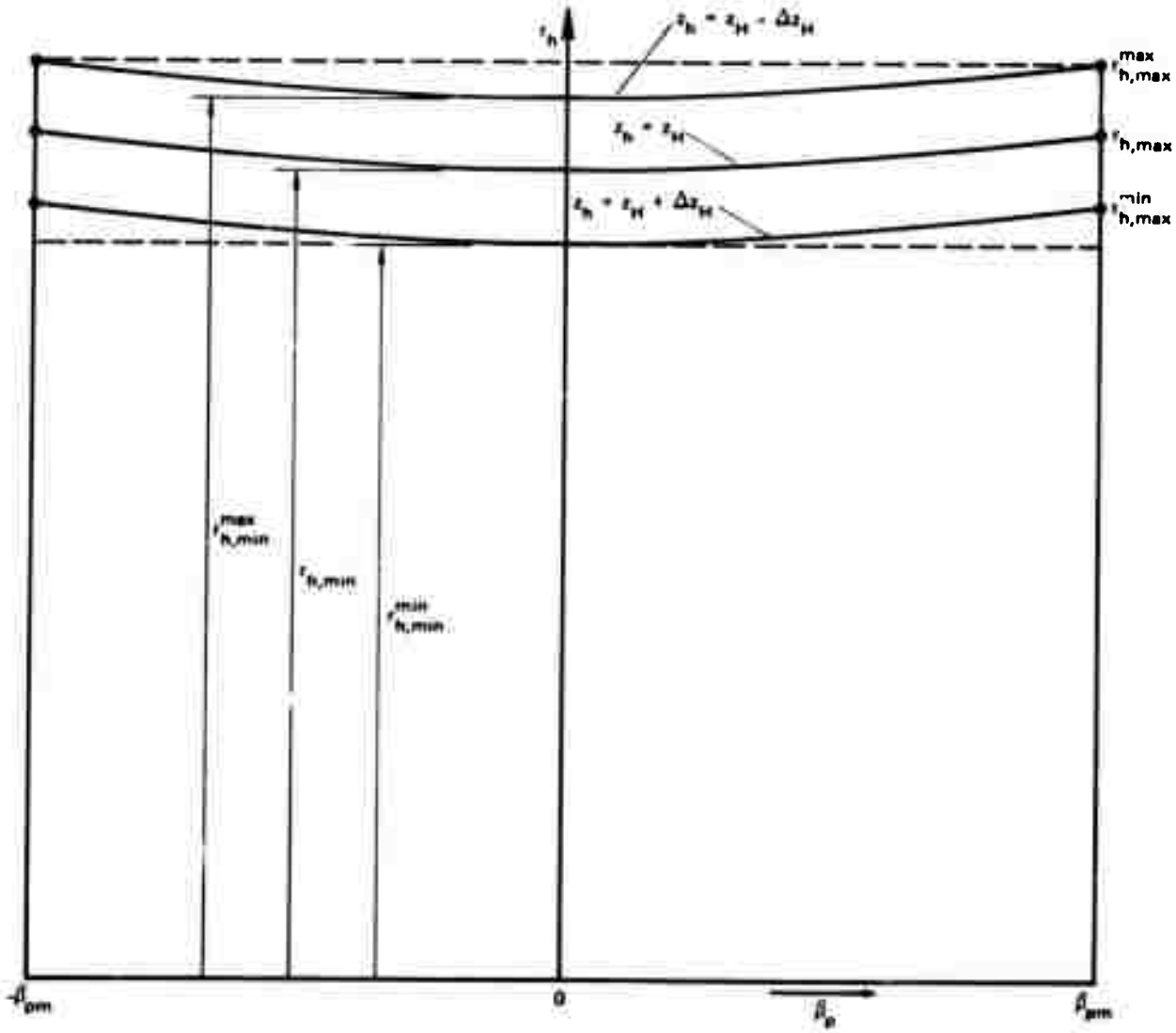
SA-1530-6

FIGURE 17 COMPUTED FLOOR BOUNDARY

"F" points agree with the true ones; "D" points differ from the true ones. Points marked by "B" indicate surface boundaries.



(a) SIDE VIEW OF A RANGE FINDER AND A HORIZONTAL SURFACE



(b) r_h VERSUS β_p

SA-1530-2

FIGURE 18 RANGE VALUES WITH CONSTANT φ_p SCAN OF A HORIZONTAL SURFACE BELOW THE HORIZON

occludes another), shadow boundary (along a region where range data are missing), known region boundary (enclosing a region, such as the floor, a horizontal surface, or a vertical surface, whose image points are given), and intersection boundary (between the images of two nonparallel adjacent planar regions). The first four types of boundary are collectively called surface boundaries because they are sufficient to outline the contour of an object surface. Surface and intersection boundaries are illustrated in Figure 19, assuming that only the image points belonging to the floor and the first step are known a priori.

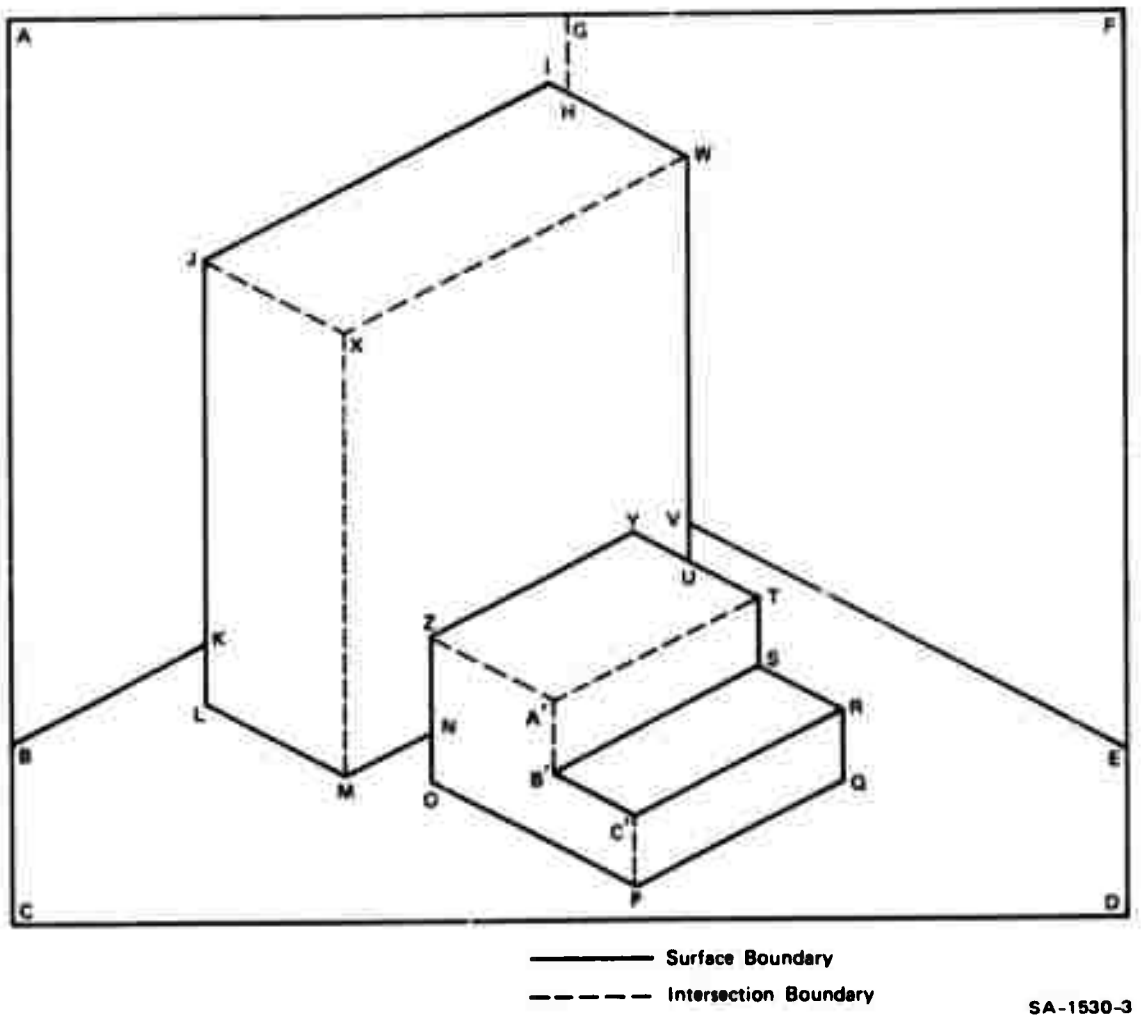


FIGURE 19 SURFACE AND INTERSECTION BOUNDARIES

The points that form the surface boundaries are easily determined: Frame and shadow boundaries are given; a known-region boundary is obtained as the perimeter of a given set of image points; a jump boundary point is established during the horizontal or the vertical scan if a range value differs from the preceding one by more than a threshold value and if the following range discontinuity is less than this threshold (the latter condition prevents confusing a true jump boundary with range increments on a very oblique surface).

Intersection boundary points are established by first assuming that adjacent pairs of surface boundary points on each horizontal raster line (of constant φ_p or z_p) bound the same planar surface. The absolute value of the error between the measured range, $r^*(\beta_p)$, and that predicted by these ideal surfaces, $r(\beta_p)$, is computed for each point in these intervals. The maximum error, Δr, equals $|r(\beta_p) - r^*(\beta_p)|$. If Δr between any pair of boundary points exceeds a threshold, then an intersection boundary is established at that point, and the process is repeated recursively for the left and right surfaces thus established. Figure 20 illustrates this fitting process for two surfaces crossing at an intersection boundary. (The x-v plane corresponds to the plane swept out by the range finder scanning a line of constant φ_p.) Notice in Figure 19(b) that we fit trigonometric functions (instead of straight lines) to the range data, because a "range image" of a straight line is not itself a straight line. The above, called the "end-point fitting method," is applicable to convex and concave polyhedral surfaces, as is shown in Figure 21, where N_i, the number of intersection boundary points, is 12. The method can also be applied to obtain a piecewise planar approximation of a curved surface. The cost of computation entailed in using the end-point fitting method is (in the worst case) proportional to N_ℓ, the highest level of recursion. It can be shown that

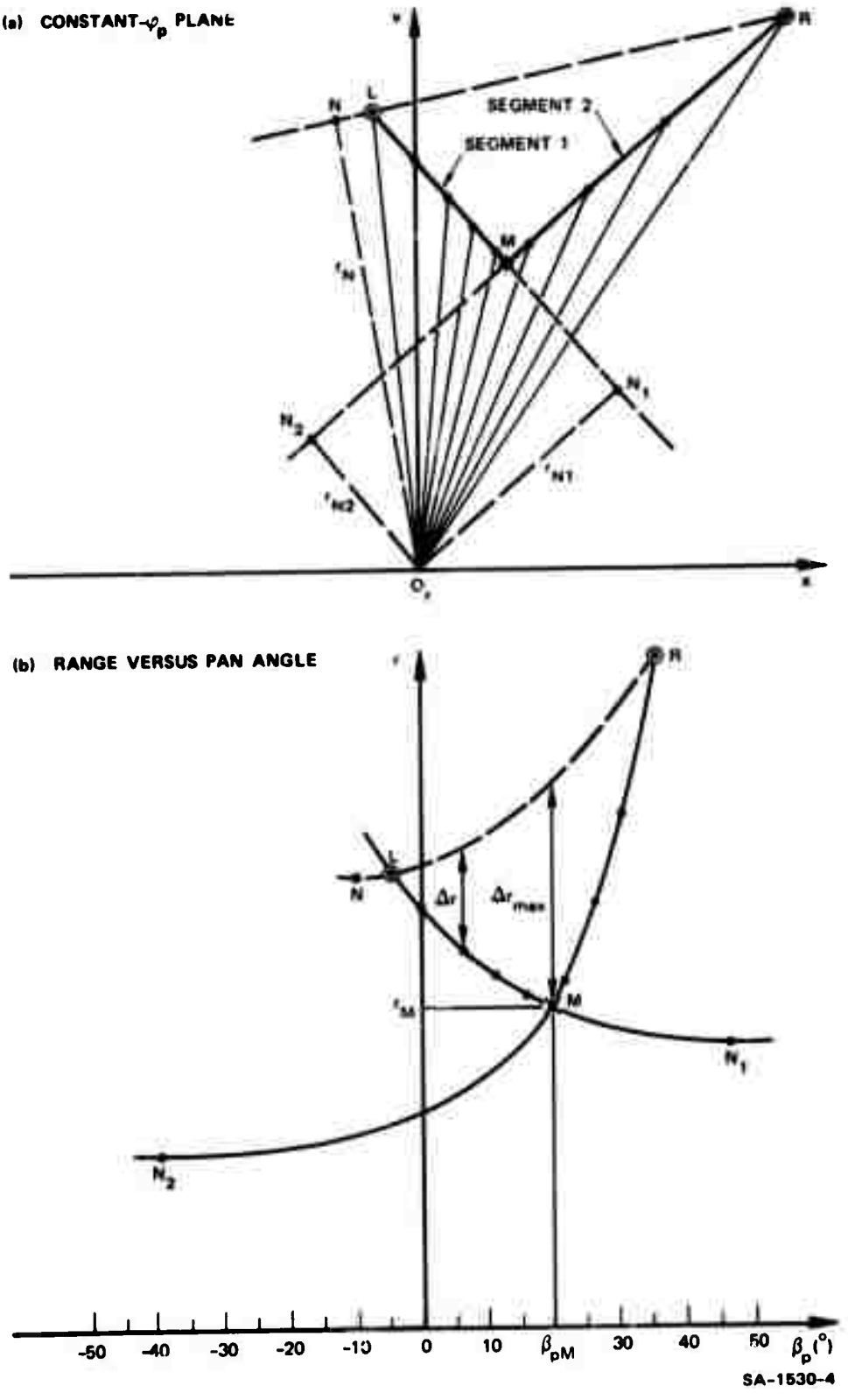


FIGURE 20 END-POINT $r(\beta_p)$ FITTING METHOD

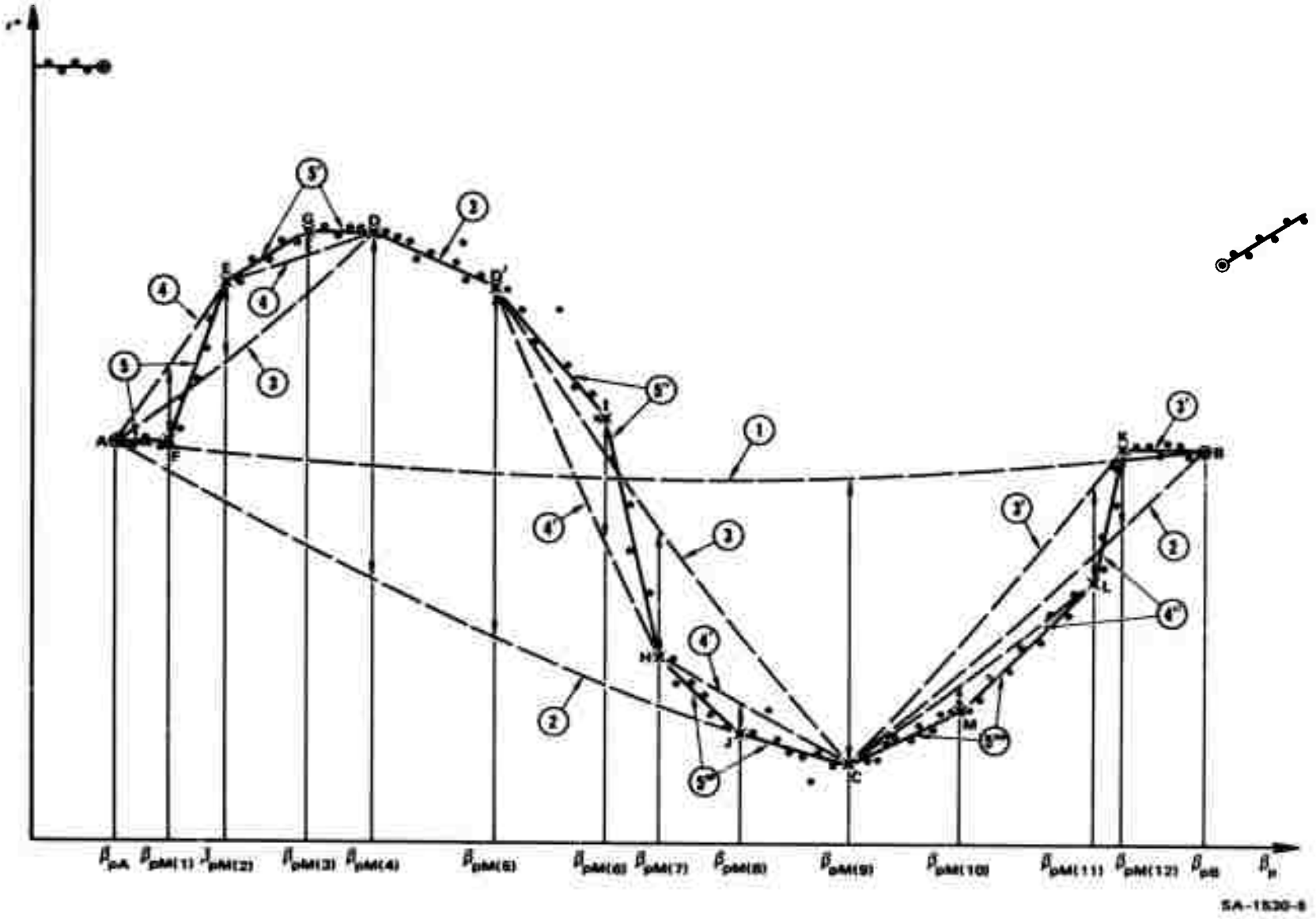


FIGURE 21 MULTIPLE $r(\beta_p)$ PLANAR-REGION SEGMENTS OF A CONCAVE AND CONVEX SURFACE

$$1 + \lceil \ell n\ (1 + N_i)/\ell n2 \rceil \leq N_\ell \leq 1 + N_i \quad ,$$

where $\lceil\ \rceil$ denotes round up to the nearest integer. Fortunately, N_ℓ is likely to be equal to its lower bound, especially if N_i is large.

2. Effects of Errors in Range Data

We now examine the effects of errors in the range data on the resulting region and boundary points, and propose some context-dependent ad hoc rules that may minimize these adverse effects. First, a slightly expanded floor region is extracted by using $\alpha < 1$. (This minimizes the error incurred in subsequently fitting non-floor planes by ensuring the exclusion of floor points.) Small adjacent regions or "islands" of no data may be considered part of a large region if there is reason to assume that they resulted from erroneous data. Jump boundaries resulting from high $r^*(\beta_p)$ "spikes" of decreased range may be disregarded if no thin objects are expected and if r^* is relatively short. Horizontal and vertical scans, in conjunction with different data errors, may result in wide intersection boundaries, but fortunately have no effect on widening the surface boundaries. Two cases are distinguished relative to $r^*(\beta_p)$ spikes: if only smooth and relatively large region surfaces are expected, then the spikes are disregarded; otherwise, the spikes are associated with the surface texture.

Errors in range data may also cause gaps in the postscan boundaries, especially the intersection boundaries. These gaps may be bridged by the line fitting methods described next.

3. Line Fitting of Region Boundary Points

Methods are proposed for determining gap-free surfaces and intersection boundaries by line fitting the boundary points found previously. We begin with vertical boundaries, because these are common

and easiest to compute. We divide the x-y plane (within the limits of interest) into L × M cells, compute the x and y coordinates of every boundary point, detect high-count cells using triple entry quantization, identify the cells with the top views of vertical boundaries, and compute the extreme z values of each vertical boundary from its extreme I values. Two types of quantization cells are considered: a fixed size rectangular cell and a variable size curvilinear cell. The latter matches the inherent error of a range finder of given angular tolerance. After deleting the vertical boundary points, we determine the horizontal boundaries. The computation is similar to that of vertical boundaries, except that we begin by detecting clusters of similar z values, compute the x and y coordinates of the high count clusters, and fit straight segments to the resulting (x,y) points. Finally, we compute the parameters of three-dimensional straight boundaries by fitting straight lines to the projections of the remaining boundary points onto the x-y and y-z planes, and compute their end points.

4. Analysis of Color Data

A color recognition module has been implemented, which takes as input the absolute intensities observed through three or more band pass filters (normally the standard red, green, and blue separations). Its output names a hue and provides a numerical estimate of the purity of that hue. This program has three distinguishing features:

- Interactively trainable--New hues are learned by example, when they are encountered for the first time. The initial characterization can be refined with additional examples if the color is later misclassified. This approach overcomes several disadvantages associated with using a fixed characterization of hues, obtained, for example, from a standard observer table. Achromatic hues are recognized by low saturation and identified as

white, gray, or black depending on their relative brightness compared with averages taken over the scene.

- Additional filters possible--Hues can be characterized in terms of an arbitrary number of filters. More filters may allow discrimination between hues that cannot be distinguished with three color separations. Narrow band color (e.g., recognizing objects by the presence of certain spectral lines) is an extreme example of how additional filters could be used to capitalize on knowledge of the environment.

- Color constancy--Illumination observed from objects is a product of incident illumination and object reflectivity. To obtain the intrinsic hue of an object it is necessary to normalize by the observed illumination to eliminate the incident component. Our system is designed to achieve first order color constancy by normalizing out the illumination source. (Second order effects on incident illumination, such as reflection from neighboring objects, are not handled.) Source normalization allows colors learned under one illumination to be recognized under any other illumination, once that source has been calibrated using an object whose color is known.

5. Generation of Test Data

The perceptual system uses arrays of both color and range data. Because we did not want to wait until the scanning rangefinder (described in Section V) was built, interfaced, and calibrated, we decided to create simulated range data. The interactive program that produces an array of range data from a picture array works as follows. A picture is taken and its gradient is shown on the CRT display. An operator controls a TV cursor to outline polygonal boundaries of the planar regions in the scene. He also identifies each region, thereby providing the program

with the parameters that define each plane in three-dimensional space.* Thus, for every point in the picture, the corresponding plane in three-dimensional space can be found by table lookup.

The procedure described by Wichman[8] and Falk[9] is used to determine a ray in space for each picture point. Their procedure uses four reference points on the floor to determine the collineation transformation between picture points and floor points, and two reference points off the floor to determine the lens center. Thus, for any picture point the collineation transformation yields the corresponding floor point, and this together with the lens center defines the ray in space. The range is obtained merely by computing the intersection of this ray with the corresponding plane. Repetition of this process for each point in the picture produces a range array in exact register with the picture array.

F. Research Methodology

In the preceding sections we have described our design for a system for interpreting perceptual data. Our belief is that such a system can be successfully implemented only if we explicitly accommodate ourselves to the empirical nature of machine perception research. Accordingly, the design includes certain key features to facilitate experimentation with real scenes and to enable us to incorporate the experimental results into an evolving system implementation. We first single out these features and then present our overall research plan.

---

*These parameters are currently measured manually. If desired, they could also be obtained interactively by having the operator designate corresponding points in two stereo views.

1. System Features

Two key features that should make our system especially valuable as a research tool are that it is data driven and highly interactive.

a. Data Driven Organization

Almost all of the system's perceptual knowledge will reside in its knowledge base, in easily accessible declarative form. This knowledge includes object attributes and their relations, as well as the cost and reliability of perceptual operators for extracting them. The system itself consists essentially of just two relatively small programs. One selects a cost-effective sequence of attributes and the other interpretively executes the resulting strategy. Very little perceptual understanding is required for either program. Thus, once they are debugged, the system can be incrementally refined and expanded primarily by modifying the symbolic knowledge base (and occasionally by programming a new perceptual operator).

b. Facilities for Interaction

The experimenter will be able to test manually generated strategies either by interactively calling system modules or by specifying the complete strategy as a program for interpretive execution. New visual concepts will be designated either by entering symbolic descriptions or, in the absence of adequate semantics, by outlining them (with a cursor) on a grey scale display.

The combined capabilities for graphical as well as symbolic communication should prove to be a powerful experimental tool for perception research. For example, specified perceptual operators can be applied interactively to graphically designated surfaces to determine how effectively they discriminate that surface from adjacent ones or

from those previously characterized in visual memory. The resulting information can then be used to select, respectively, good region growth and acquisition criteria. The outlined surface can also serve as an exemplar for learning new colors, textures, shapes, and the like, "by example." Symbolic values for these attributes (e.g., red, grainy, elongated) can be represented internally in terms of perceptual operators that discriminate the designated surface from exemplars of previously defined values. Finally, structural models can be constructed interactively by specifying spatial constraints between outlined parts of an object. The constraints can be provided symbolically (e.g., part A must be above part B) or photogrammetrically, by specifying measurement routines to extract metric relationships designated graphically in the picture.

2. Plans

We plan to implement the system in, roughly, four phases.

Phase 1: A skeleton system is now being assembled, consisting of some primitive perceptual operators, an interactive interpreter for applying those operators, and a facility for graphical communication. This facility will initially be used to validate experimentally our basic premise that there exist easy ways of distinguishing objects in constrained contexts. The investigator will attempt interactively to generate distinguishing features strategies (like those in Section III-B). for finding the various scenario objects in a few test scenes. Criteria for acquisition, growth, region evaluation, and validation will be selected pragmatically, by choosing the most distinctive attributes of an object that are easily represented in terms of available perceptual operators. Immediate feedback from the system will indicate the validity of these intuitions.

Strategies will occasionally fail because the investigator had not anticipated the literal implications of applying a single criterion to all objects. The investigator may then specify more constraining criteria. However, we expect that failures will more commonly arise due to inadequate primitives that cannot reliably perceive intended discriminations. In such cases, one might again choose other criteria, but if no suitable alternative exists it becomes necessary to refine the programmed operators. We expect a set of adequate operators to emerge as a major result of this phase of research.

Successful perceptual strategies constitute procedural descriptions of objects, and will be retained to establish the system's initial world model. The corresponding initial planning algorithm for finding an object is then simply to retrieve the appropriate strategy.

Phase 2: In this phase our emphasis will shift to machine generated strategies for finding a limited set of objects, efficiently as well as reliably, in a wide range of scenes. To this end we will augment the knowledge base with descriptions of objects in declarative form and will develop basic modules for planning and execution monitoring, like those outlined in Section III-C.

Progress in Phase 2 depends in good part on obtaining quantitative expressions for utility and recognition confidence. We seek, for each object feature, numerical estimates of utility that correlate reasonably well with the effectiveness of corresponding strategies. We also require pragmatic expressions for recognition confidence, so that we can terminate our sequential decision process on the basis of confidences observed so far.

We will not be concerned in this phase with planning efficiency, but will instead search exhaustively for the best plans. Objects will

be described in the knowledge base directly in terms of their distinguishing features. These features, in turn, will be described in terms of specific feature extraction operators. This allows us to defer to a subsequent phase of research the key problem of automatically discovering good distinguishing feature representations for objects.

We expect to arrive empirically at both adequate object descriptions and utility estimates by evolving them from the initial intuitive estimates developed during Phase 1. A human experimenter will analyze failures of machine generated strategies and interactively refine utility estimates. He will also be able to enhance discrimination by suggesting additional object attributes and/or contextual constraints. Interactive refinement of crude intuitive estimates should prove an effective compromise between the unworkable extremes of unsupervised learning and exhaustively analyzed initial models.

When strategies can be generated for reliably finding the original objects, the scenario will be incrementally enlarged to encompass additional office items. Descriptions will again be interactively generated. The experimenter will suggest initial features that distinguish the new object from others already in visual memory. Those features may then be refined with feedback on their empirical effectiveness.

Phase 3: Phase 3 will concentrate on developing heuristics for efficient planning. The effectiveness of strategies generated under various simplifying assumptions will be compared with the effectiveness of those generated by exhaustive search in Phase 2. Two representative heuristics that will be studied are the use of a fixed budget percentage as a termination criterion, and the elimination of combinatorics introduced by the consideration of shared modules.

Phase 4: In this final phase we will attempt to automate much of the decision making and problem solving handled interactively in Phases 1 and 2. Specifically, the system itself should deduce from its general fund of world knowledge the features and corresponding representations that best distinguish a given object in a particular context. The data base might now include knowledge about the functions and material compositions of objects, models of perceptual operators and their error processes, and the level of detail needed to fulfill various task objectives.

Phase 4 is a formidable long range goal that may well require breakthroughs in automatic programming. However, we feel that perceptual strategies constitute a particularly well structured domain in which to explore the key issues. In fact, we suspect it will be again possible to proceed incrementally, by adding knowledge and inference rules, as necessary, to the basic Phase 2 system.

IV SYSTEM SOFTWARE

A. Introduction

Several features were needed at the close of the previous year in order to enhance the utility of the robot as a general purpose experimental vehicle. This section discusses developments on this front from a software viewpoint; as will be explained presently, we have made considerable progress. For the purpose of discussion, we can view the process of increasing the experimental utility as proceeding along two paths: making the robot easier to use, and telling people how to use it. Here we describe the former task; a recently completed primer[10] describes the current robot system from a user's viewpoint.

There were three parts to the task of making the robot more usable. First, we made the robot usable to an experimenter who has a minimum of specific knowledge of its inner workings and second, we greatly increased the speed of the robot so that the experimenter would not be thwarted by lack of extraordinary patience. Finally, we made the overall robot action system more reliable. Now, for example, the robot can execute a fairly long STRIPS-produced plan with reasonable assurance of success in the face of normal day-to-day radio interference and other adversities.

B. Utility Factors

1. Comprehensibility

We have increased the comprehensibility of the robot action routines in two ways: we have made them available in a neat, self-contained package, and we have included in this package a command interpreter. Formerly, we had provided the action routines primarily in

symbolic (source code) form for the would-be user (necessarily a programmer) to include with his own software. Now the primary form in which the action routines are available is a LISP SYSOUT file, which makes the action package almost as easily available as any other major subsystem within TENEX. We now have a command interpreter (which was easily provided as a minor extension of an already existing debugging aid). This modest amount of software permits a novice experimenter to use the action routines as programs, and to easily obtain informative status reports couched in meaningful terms. The combination permits a person to establish contact with the robot in less than a minute by typing three command lines.

2. Speed

The action routines in their most primitive state were executed so slowly as to require inordinate patience on the part of the user. Around the beginning of the project year, we changed from DEC's time-sharing system to BBN's TENEX time-sharing system. This gave us a tremendous speedup due to the difference in swapping techniques; formerly, we had to swap a program's entire core image into core before running it, whereas we now can swap only a small part of a program into core before running it. The gain was especially great because a typical job mix contains several programs each large enough nearly to fill user core. At the beginning of the project year, then, a single complex ILA (intermediate level action) could still take as long as 40 minutes to execute. At that stage, the action routines were run entirely in an interpretive mode. The speedup techniques we employed were a combination of compiling and rewriting the routines and trasnferring the whole package to BBN LISP. The net result of our efforts was to reduce execution times of the more complicated ILAs by a factor of about five.

3. Reliability

The reliability improvements we made are an extension of the ruggedness already built into the software. For example, the ILAs have always been capable of coping with accumulated small mechanical inaccuracies, or with more serious events such as encountering unexpected obstacles. This year we extended the ruggedness to an ability to cope with more drastic failures, such as a telemetry failure. Provision is made for error handling in many places throughout the action routines. Each routine is rather limited in scope, and so the overall ruggedness is due more to the dovetailing of the error provisions than to the power of the error handling in any one of them.

One of the more interesting illustrations of this ruggedness was the following incident. During an experiment, the robot lost power in mid-maneuver; power was hurriedly restored to the robot, which without further intervention visually ascertained its position and proceeded in its execution of the ILA in progress at the time. The loss of power control caused the LLA's to report a large uncertainty in the robot's position. The response of the ILAs was to attempt (multiple times, as is their way) to ascertain the robot's position visually. Finally succeeding, the normal (for the ILAs) reassessment of the goals and the information at hand led the ILAs to continue the maneuvers in progress. The reliability features of the robot software are greatly enhanced (if not made possible) by the flexible error handling features of the BBN LISP system.

C. Translation to BBN LISP

1. Overview

For reasons outlined elsewhere in this section, we converted the robot action routines from the Stanford LISP system to the BBN LISP implementation. This translation was tremendously aided by TRANSOR, a

BBN-supplied translating program. We added a few programs to the TRANSOR system to automate additional required or helpful ancillary processes, and we tailored the transformations to the purpose at hand. TRANSOR made all straightforward transformations automatically; it noted all doubtful constructs and transformed them as well as it could; and it especially flagged all constructs requiring hand translation. The principle part of the action software requiring hand work was the support package, which handles symbolic disk files containing STRIPS operator descriptions and the ILA routines (in the form of modified Markov tables). This support package had been well integrated into SMILE, the Stanford LISP package for handling symbolic files. We now have attached it to BBN's "pretty-print" package, which serves the same function in the BBN implementation. The transfer of the support package required considerable hand work since it was outside the useful scope of TRANSOR. However, this hand work afforded us the opportunity to rework this relatively new software and not only weed out the vestiges of the usual false starts, but also to increase its flexibility and coherence.

2. Forking

To date we have transferred the individual pieces of the robot system to BBN LISP. What remains is the reestablishment of the links from the action routines to the STRIPS/PLANEX cluster on the one hand and to the vision routines on the other. Part of the price of the high degree of flexibility offered by the BBN LISP implementation is a reduction in the memory size available to a user's own program. In our case, the available address space was large enough to accommodate either the STRIPS/PLANEX cluster or the action package alone, but it was not large enough to accommodate them both. A more fundamental restriction of the BBN LISP system, however, is its inability to accommodate FORTRAN (or other non-LISP) programs; the Stanford LISP system provided a usable, if

awkward and inflexible, means of accommodating such "foreign" programs. There is, however, a boon in BBN's TENEX time-sharing system: forking. That is, any program can have some number (about 25) of subprograms, and each of those programs (the mother as well as her daughters) can have completely independent address spaces of one-quarter million words each. Other projects within SRI's Artificial Intelligence Center have jointly developed software to handle the interfacing of forks with their descendants (and ancestors), including creation and other initialization capabilities. This software permits a LISP program to generate multiple independent subforks and to call FORTRAN subroutines within them. We intend to adapt this software to permit the subfork to be LISP and/or to call subroutines in its mother. The former will permit PLANEX actually to call the action routines, and the latter will permit the vision routines to move the robot via the action routines (instead of requiring the vision routines to have their own action routines).

3. Additions for TRANSOR

Before we leave the subject of the transfer to BBN LISP, we will briefly describe our addition to BBN's TRANSOR. As supplied, it consisted of four parts: a rule driven translator, a set of rules, a package of programs to generate or modify such a set of rules, and a prescanner to make certain simple character transformations so the translator could read the program to be translated. Of these, we extended the power of the set of rules (mainly to translate our modified Markov tables) and the prescanner (to cope with the way Stanford LISP handles overflow of a print line). This still left two of the conventions of Stanford LISP with which TRANSOR could not cope: (1) Stanford LISP could (and in practice often did) use octal numbers (instead of decimal), and (2) it also could (and often again did in practice) flag decimal numbers as such by following them with a decimal point. The BBN

LISP system (in which the translator was embedded) would interpret an octal number as the decimal number with the same sequence of digits, and it would interpret the trailing decimal point as a sign to represent the number internally in floating-point format. However, the Stanford LISP system is also perfectly capable of producing numbers in a BBN-compatible format, so we simply wrote a small "pre-prescan" program in Stanford LISP that would read a Stanford LISP file in any format and write it in a format suitable for the BBN system. This gave us a system in which a file of Stanford LISP programs was first "pre-prescanned," then prescanned and then translated; this produced the translated version as well as a file of notes pointing out dubious portions of the output file. This last had to be listed so the programmer could check the translation; then it and the two intermediate files from the prescanners had to be deleted. Typically, the next step was to load the translated file for checkout and any necessary hand modification. We wrote a small program to perform all these steps and to produce a translated version given only the name of the Stanford LISP version of the file.

V SYSTEM HARDWARE

A. Introduction

In this section we briefly describe several pieces of special purpose hardware that we are currently building or interfacing to our robot system. This hardware includes two types of scanning range finders, a doppler radar (for detecting motion) and a Unimate arm. We also briefly describe the current configuration of our complete computer system.

B. A Time-of-Flight Range Scanner

1. Introduction

For the purpose of scene analysis in a three-dimensional world, it is desired to generate a "picture" analogous to a television picture, but where each picture point is associated with an analogue value denoting range rather than light intensity. The simplest method for determining the range of each picture point might seem to be by measurements on stereo pairs of images taken by the same television camera. In practice this method has proved to be seriously limited on at least four counts:

- Inadequate resolution: at the 120 by 120 element resolution we have been using, the displacement (measured in elements) between corresponding points in the images is not enough.
- The range accuracy falls off rapidly with distance.
- The calculations for so many points consume too much computer time.
- There are difficulties due to occlusion, because a picture element visible from one lens viewpoint may not be visible from the other.

A scanning range finder currently under construction eliminates all of the above problems in that it possesses high range resolution that does not vary with distance, and gives direct readout without computation. In addition, since it is a "one-eyed" device, the occlusion problem does not arise.

2. The Experimental Model of the Range Scanner

A simplified block diagram of the range scanner is given in Figure 22 and a more detailed diagram in Figure 23. The range scanner consists of three functional components:

- A transmitter that amplitude modulates a light beam with a 9-MHz sine wave
- A mirror scanner that sweeps the modulated beam over the field of interest
- A receiver that picks up the light from whatever object intercepts the scanning beam. After amplification the output from the receiver is demodulated by a phase demodulator.

The demodulator effectively measures the length of time required for a light beam to make the round trip from range scanner to light scattering object and back. The range value, of course, is a constant multiple of this time.

The merits of the phase demodulator are worth emphasizing. In this device the output from the receiver is compared with the phase of the modulating signal feeding the transmitter, and a dc level is derived that is a function only of this phase difference. Since the phase difference varies linearly with the distance to the light scattering object, the dc level is a measure of the range of the object. The method has the virtue of being a one-frequency method not depending on the amplitude of the received signal; it is therefore possible to trade range discrimination

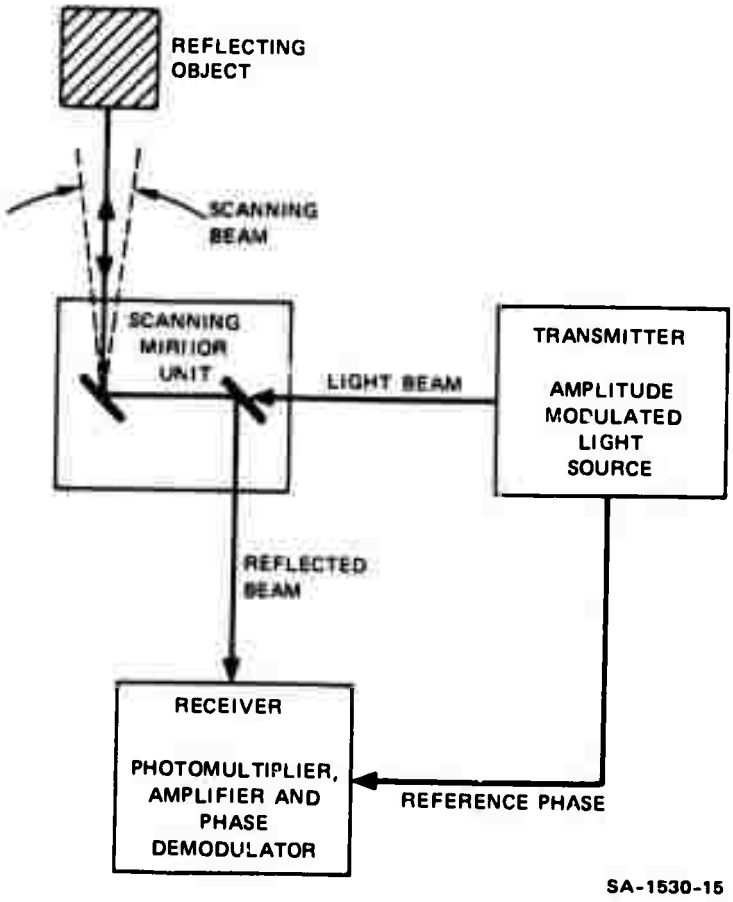


FIGURE 22 SCANNING RANGE FINDER—SIMPLIFIED

against signal integration time. The sensitivity calculation described later indicates that it should be possible to make 1,000 measurements per second with a range discrimination of 0.1 inch using a transmitter power of only 1 milliwatt.

Considering the transmitter section, the process begins with the 9 MHz crystal oscillator that drives the modulator via an adjustable phase shifter. The phase shifter provides an adjustment of approximately ± 60° so that one may adjust for a precise zero or, alternatively, off-set the minimum range and use increased sensitivity in the read-out. The zero may also be displaced by fixed amounts and with great precision by short lengths of coaxial cable inserted between the various units. The

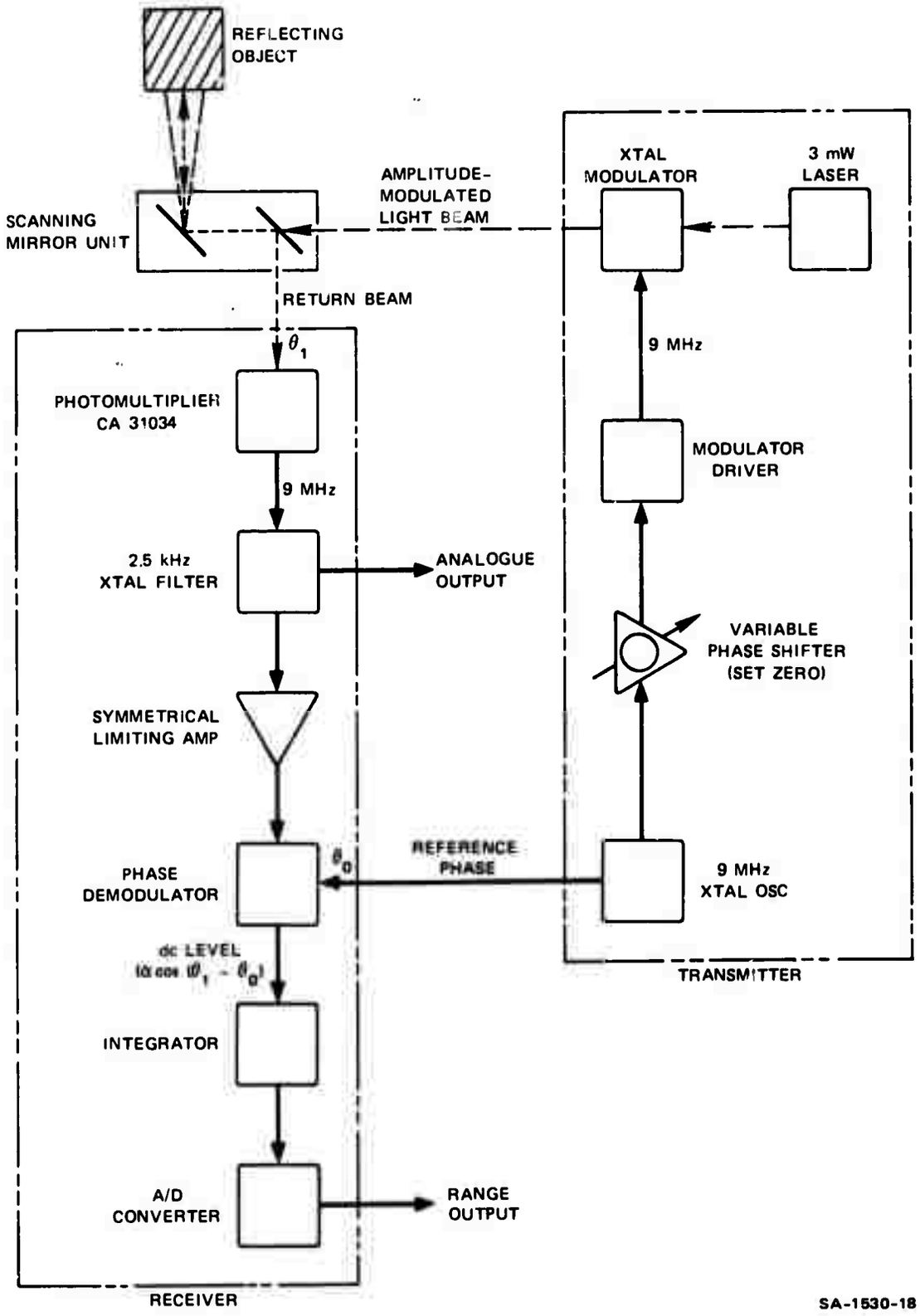


FIGURE 23 SCANNING RANGE FINDER—DETAILED

modulator then provides amplitude modulation of the light source, which may be either a 9050 A.U. gallium arsenide diode or a 6328 A.U. helium-neon laser. The latter will most probably be used since it can provide a smaller spot size (better lateral resolution). Although gas lasers typically show 1 percent amplitude noise modulation, this does not interfere, except to a minor extent, with the precision of the phase measurement. The modulated light beam is then deflected by a scanning mirror system (in two orthogonal directions) so that it scans the field of view that is of interest.

The receiver section makes use of the same scanning unit so that its line of sight coincides with that of the laser beam. The return signal is put through an interference filter to eliminate the ambient illumination and passes to a high-grade photomultiplier, an RCA CA 31034, where it is converted to electrical form. The use of a premium quality photomultipler is warranted since it is here that the signal-to-noise ratio of the system is determined. The primary requirements are for an efficient photocathode and low dark current. It is anticipated that the signal level from the CA 31034 will be at least 1 millivolt. This signal is then put through a 2.5 kHz wide crystal filter (form factor 2:1) at the first opportunity in order to minimize effects due to electrical interference, in particular before any saturation can occur. The filter is followed by a symmetrical limiter to remove amplitude variations due to inverse square law, orientation of the surface, reflection coefficient, laser noise, and so on. Note that the limiter must limit the top and bottom halves of the sine wave symmetrically since asymmetry usually introduces a shift in phase.

Perhaps the most stringent part of the design occurs in the phase demodulator. What is required is a clean multiplier. We wish to multiply together the reference signal, $e_1 \sin \omega t$, and the reflected signal that has passed down the receiver chain, $e_2 \sin(\omega t + \varphi)$. We do not have a

great deal of control over the various phase shifts that occur down the receiver chain, but we assume they can be nulled out by the phase shifter mentioned previously. We are interested in developing a dc level that varies linearly with the variable part of φ; i.e., the part due to the variable path length between the deflecting mirrors and the reflecting object. The output, M, from the multipler is

$$M = k \cdot e_1 e_2 \cdot \sin \omega t \cdot \sin(\omega t + \varphi)$$

$$= ke_1 e_2 \cdot \sin \omega t \left\{ \sin \omega t \cdot \cos \varphi + \cos \omega t \cdot \sin \varphi \right\}$$

$$= ke_1 \cdot e_2 \cdot \left\{ \cos \varphi \ \frac{1}{2} - \frac{\cos 2\,\omega t}{2} + \frac{\sin 2\,\omega t}{2} \right\} ,$$

where k is an arbitrary constant. The terms in 2 ωt are easily removed by a low-pass filter leaving an output depending only on cos φ; we shall be working over the linear region either side of 90°, as shown in Figure 24. Note that the output also varies with e_1 and e_2, and therefore these must be kept constant to at least the precision of range measurement. In the hardware that has been constructed, the output fluctuation is of the order of ±3 millivolts for a working output range of ±3 volts.

3. Calculation of Sensitivity

The initial design assumptions are as follows:

Scanning transmitter power (collimated beam) = 10^{-3} watt

Maximum range = 10 ft

Minimum reflectivity = 1 percent

The reflected power is assumed to be scattered uniformly over a hemisphere.

Receiver capture area = 1 sq in.

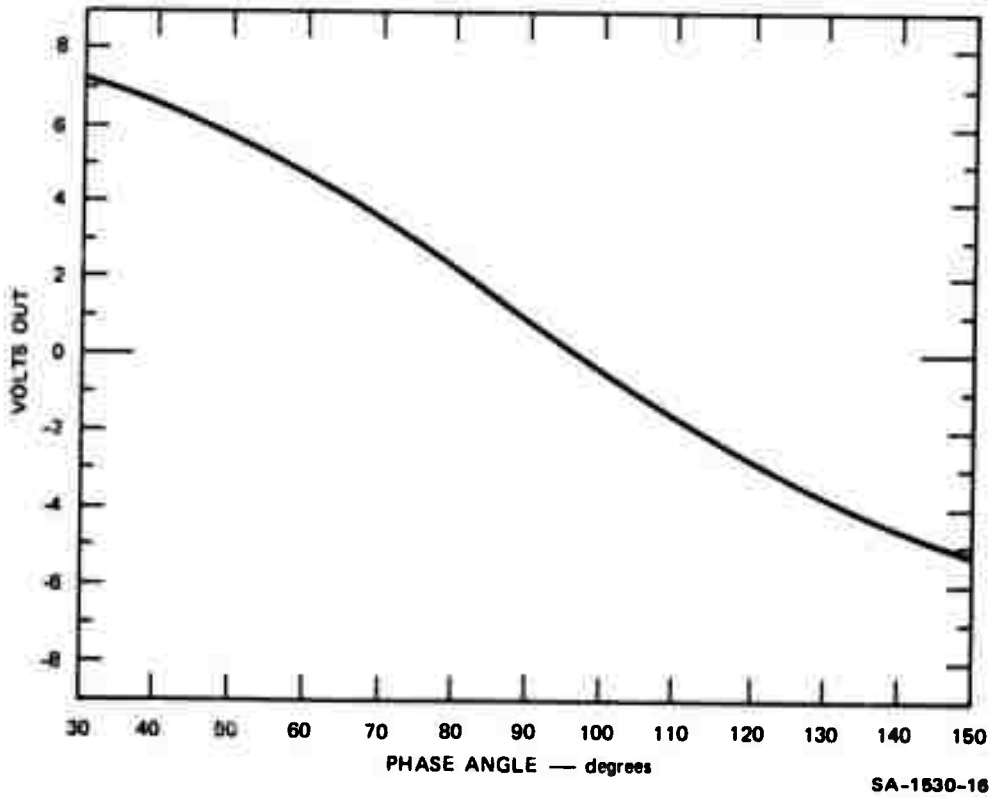


FIGURE 24 ANALOG OUTPUT FUNCTION

The received power, P, applied to the photomultipler is

$$P = 10^{-3} \cdot 10^{-2} \cdot \frac{1}{2\pi \cdot (120)^2} = 10^{-10} \text{ watt} \quad .$$

The sensitivity of the RCA CA 31034 photomultipler is given as 4×10^4 amps per watt at 6328 A.U., and 1×10^4 amps per watt at 9050 A.U. For an input of 10^{-10} watt at 9050 A.U., the output current = $10^{-10} \times 10^4$ amps, which equals 10^{-6} amps. Assuming a load impedance of 2,000 ohms

$$\text{Output level} = 2 \times 10^3 \times 10^{-6} = 2 \text{ millivolts} \quad .$$

For a bandwidth of 2.5 kHz in the receiver chain, the self-generated noise level should be less than 2 microvolts; this is compatible with 10-bit accuracy for the range output and 1000 measurements per second.

C. A Triangulation Range Finder

A range finder based on triangulation was designed and built by the late John H. Munson. A schematic drawing of this range finder is shown in Figure 25. A collimated light beam scans the field of view by means of a rotating "transmitting" mirror. A portion of this beam, scattered or reflected from a surface "point" of an object in the scene, is accepted by a rocking "receiving" mirror. The angular velocity of the transmitting mirror is much higher than that of the receiving mirror. The functions that relate time, t, to the angles φ_T and φ_R of the transmitting and receiving mirrors, respectively, are known. The cyclic time of the receiving mirror is divided into N time intervals, where N is the number of points scanned horizontally. For each of these time intervals, $\varphi_R(t)$ is known and may be regarded as fixed relative to the time varying $\varphi_T(t)$. By measuring the time of acceptance of the scattered light by a receiving photo detector, $\varphi_T(t)$ is computed. Knowing φ_T, φ_R, and the distance between the transmitting and receiving mirrors, the range (defined as the distance between the receiving mirror and an object point) is computed for each of the horizontally scanned N points. This procedure is repeated as the tilt angle of the whole system is varied incrementally by means of a tilting drive. A raster of range values for the entire field of view is thus obtained.

Although range data have been collected by this range finder, we have decided to concentrate our efforts on the range finder described in the preceding section, because the range finder of Figure 25 has the following disadvantages:

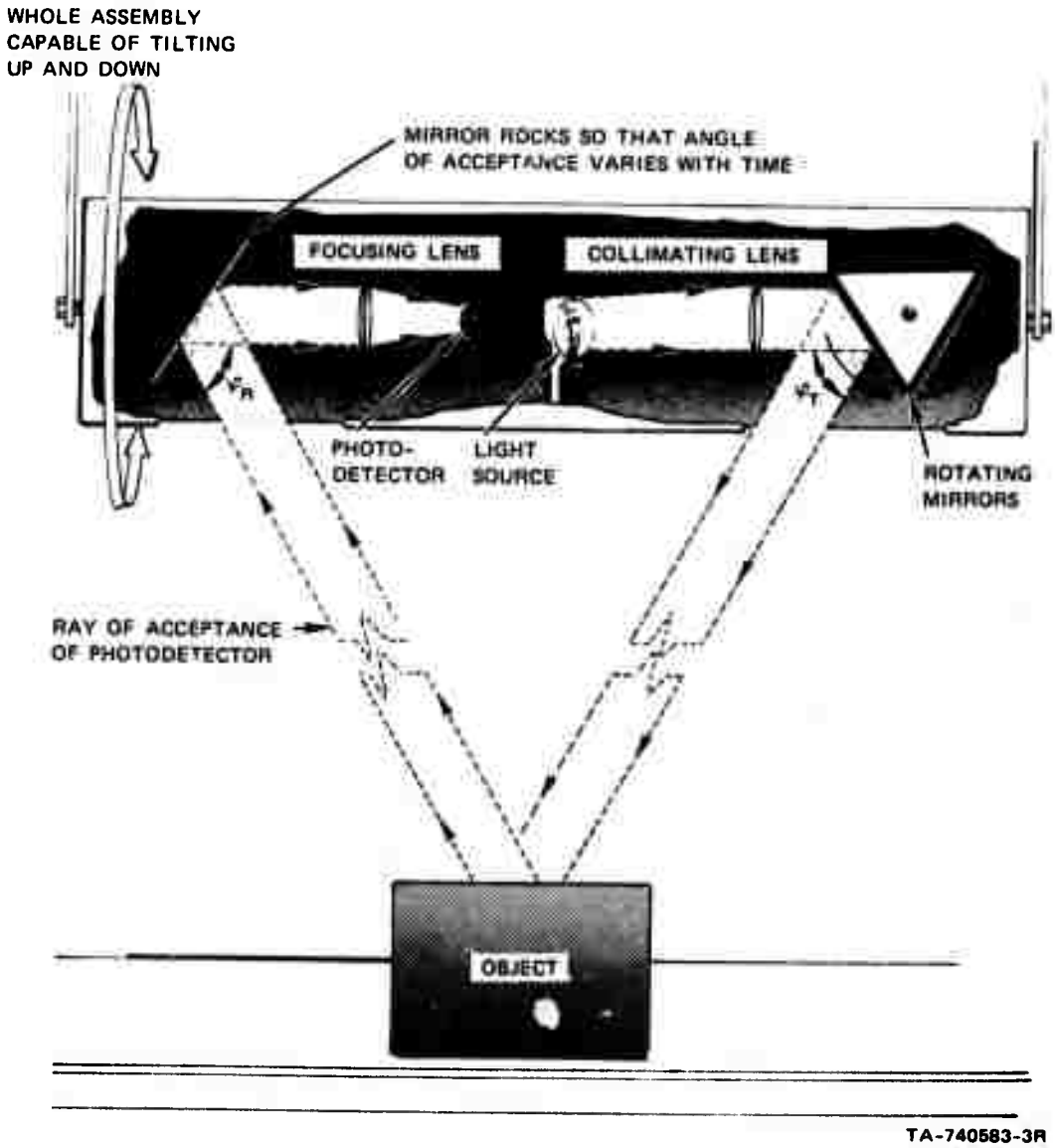


FIGURE 25 TRIANGULATION OPTICAL RANGE FINDER

- The accuracy and sensitivity are inadequate, especially as the range increases.
- Range data are missing for those object points that are seen by the receiving mirror but not by the transmitting mirror.
- The maximum range is limited to 8 feet by the intensity of the transmitting light source.
- The offset between the receiving mirror and a camera used in scene analysis poses a registration problem.

D. A Radar Motion Detector

A small, inexpensive, doppler radar, motion detector was physically installed on the robot vehicle during the summer. Its primary commercial application is as an intrusion alarm. This sensor, when made operational, can detect even a slight motion that occurs within a teardrop shaped region 40 feet wide and 50 feet long directly in front of the robot. The device is essentially a microwave transmitter radiating from a parabolic-section horn antenna and a receiver operating from a second similar antenna. The FCC allocated frequency is 10,525 mHz ±25 mHz.

This novel capability will provide the robot with a way of dealing with dynamic changes in its environment and, when interfaced, may be the basis for a new series of experiments. For example, an interrupt from the motion detector could be used to turn on the TV camera in order to determine what changes have taken place.

E. Unimate Arm

A Unimate industrial arm is being interfaced with the PDP-15/PDP-10 time-sharing system to provide a facility for multiple effector experiments. As originally manufactured, the Unimate arm and hand assembly is hydraulically positioned under the control of a program stored on a magnetic drum. The absolute position for each of five degrees of freedom is determined by optical shaft encoders. The Unimate is also conveniently

equipped with input lines to sense external events and output lines to provide indication of completed internal events

In converting the Unimate to computer control, the memory drum is replaced by a buffer register that is connected to the computer through an interface. A block diagram of the interconnections for a PDP-15/Unimate system is shown in Figure 26. A block diagram of the interface control logic is shown in Figure 27.

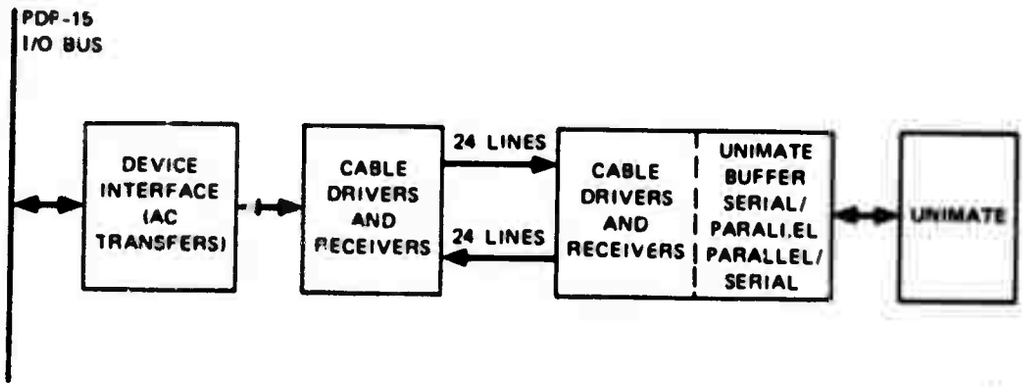


FIGURE 26 PDP-15/UNIMATE COMMUNICATIONS: BLOCK DIAGRAM

F. Current System Configuration

The current configuration of our computer system is shown in Figure 28. The primary changes made during the past year have been the addition of a fifth disk pack drive, two magnetic tape drives (and interface), a scan converter (that will allow us to display gray-scale digital pictures), and a new television camera suitable for obtaining color information.

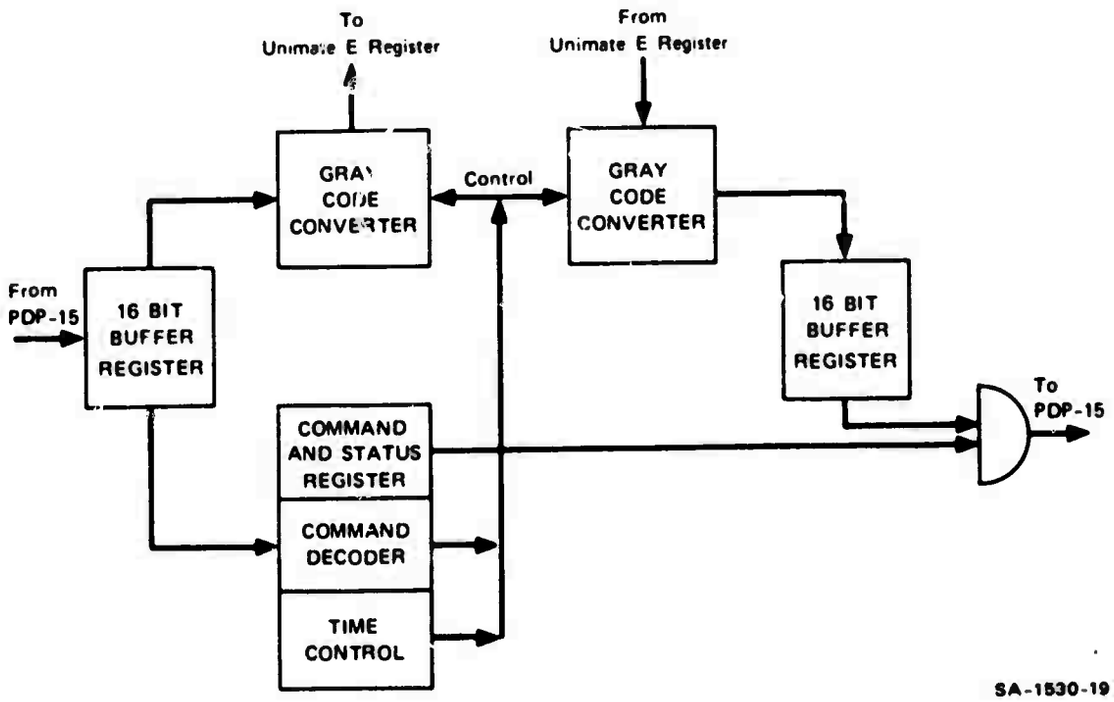


FIGURE 27 PDP-15/UNIMATE COMMUNICATIONS: UNIMATE CONTROL LOGIC

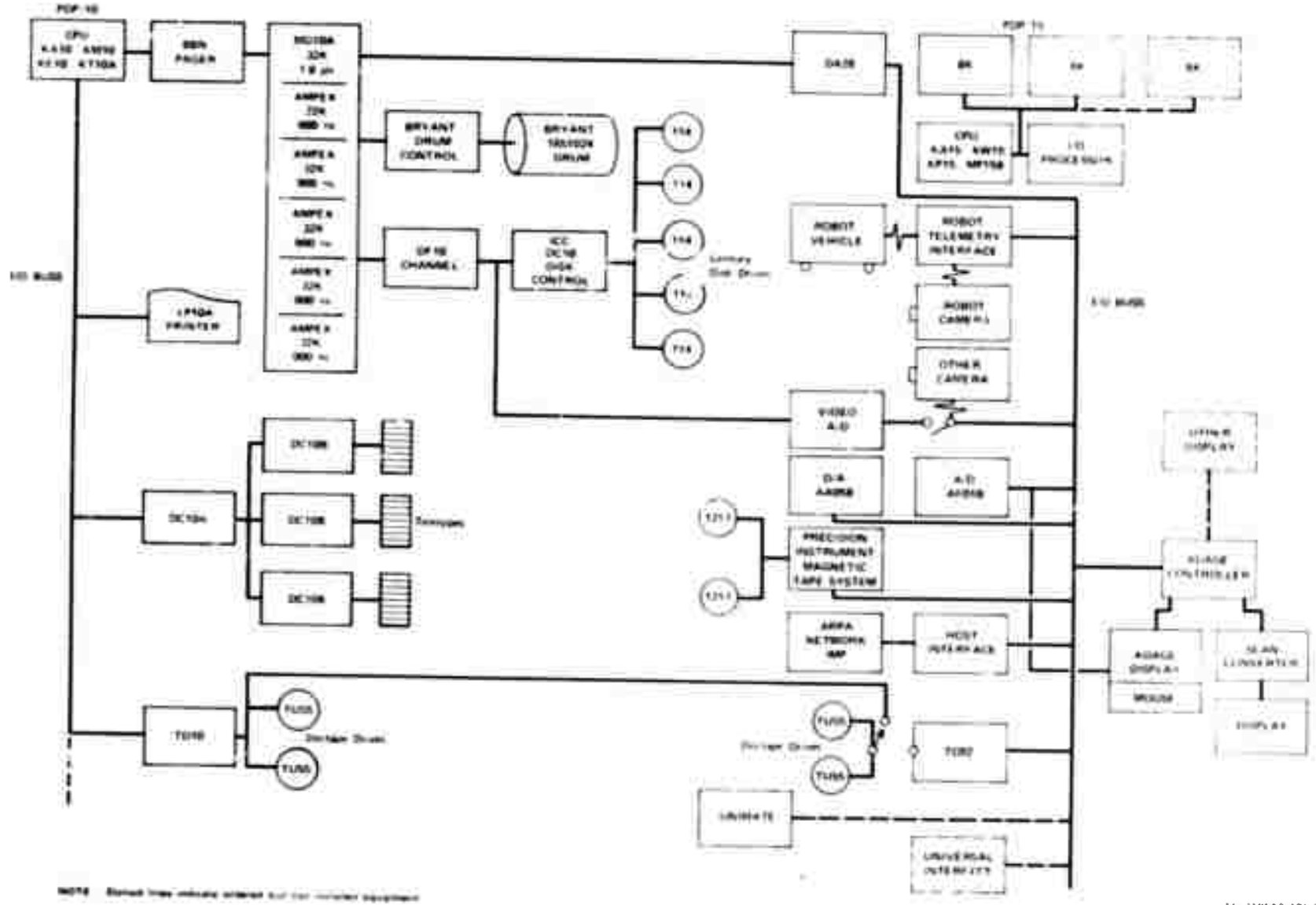


FIGURE 28 SRI ARTIFICIAL INTELLIGENCE CENTER COMPUTER SYSTEM

VI PUBLICATIONS AND PRESENTATIONS

A. Publications

B. Raphael, "The Role of Formal Theorem Proving in Artificial Intelligence," Artificial Intelligence Center Technical Note 63, Stanford Research Institute, Menlo Park, California (February 1972). This was the first of two lectures presented at the Japan Industrial Technology Association International Symposium on Pattern Information Processing Systems, Tokyo, March 6-17, 1972.

_____, "Robot Research at Stanford Research Institute," Artificial Intelligence Center Technical Note 64, Stanford Research Institute, Menlo Park, California (February 1972).

This was the second of two lectures at the Japan Industrial Technology Association International Symposium on Pattern Information Processing Systems, Tokyo, March 6-17, 1972.

R. Fikes, P. Hart, and N. Nilsson, "Some New Directions in Robot Problem Solving," Artificial Intelligence Center Technical Note 68, Stanford Research Institute, Menlo Park, California (May 1972).

Also to be published in Machine Intelligence 7.

_____, "Learning and Executing Generalized Robot Plans," Artificial Intelligence Center Technical Note 70, Stanford Research Institute, Menlo Park, California (July 1972).

Also will be in Artificial Intelligence, Vol. 3, No. 4 (to appear Winter 1972).

D. Nitzan, "Stereopsis Error Analysis," Artificial Intelligence Center Technical Note 71, Stanford Research Institute, Menlo Park, California (September 1972).

_____, "Scene Analysis Using Range Data," Artificial Intelligence Center Technical Note 69, Stanford Research Institute, Menlo Park, California (August 1972).

P. E. Hart and N. J. Nilsson, "Shakey: Experiments in Robot Planning and Learning," 25-minute color and sound film that reviews recent experiments, Stanford Research Institute, Menlo Park, California (August 1972).

B. Presentations

Technical meeting on Computer Vision held at Pajaro Dunes, California, November 10-13, 1971. Attendees from SRI: T. D. Garvey, C. A. Rosen, and C. L. Fennema.

N. J. Nilsson, seminar talk entitled "Composing, Using, and Executing Robot Plans" at the University of Illinois, Champaign, Illinois, December 16, 1971

Firbush Point Seminar held at the University of Edinburgh, Scotland, February 28 to March 3, 1972. Attendees from SRI: R. E. Fikes, P. E. Hart, N. J. Nilsson, and J. F. Rulifson.

N. J. Nilsson, lecture tour in 1972: University of Calgary, Alberta, Canada, March 22; University of Alberta, Edmonton, March 23; University of British Columbia, Vancouver, March 24. The talk was entitled "A Problem-Solving System for a Robot."

Current Research in Problem Solving and Machine Perception Seminar held at the University of California, Department of Psychology, San Diego, California, April 17, 1972. Attendees from SRI: R. E. Fikes, B. Raphael, and J. M. Tenenbaum.

R. E. Fikes, seminar talk entitled "New Experiments with Shakey the Robot" at the University of California, Irvine, California, June 2, 1972.

P. E. Hart, seminar talk entitled "A Short Survey of Artificial Intelligence and Robots," at Jet Propulsion Laboratory, Pasadena, California, June 30, 1972.

C. A. Rosen, seminar talk entitled "Robots, Productivity, and Quality," Jet Propulsion Laboratory, Pasadena, California, June 30, 1972.

J. M. Tenenbaum, talk on robotics to ACM Peninsula Chapter, July 13, 1972.

Remotely Manned Systems Conference, Pasadena, California, September 13-15, 1972; sponsored by the National Aeronautics and Space Administration and the California Institute of Technology. Attendees from SRI: P. E. Hart, N. J. Nilsson, and C. A. Rosen.

REFERENCES

1. R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," Artificial Intelligence, Vol. 3, No. 4 (to appear Winter 1972).

2. P. E. Hart and N. J. Nilsson, "Shakey: Experiments in Robot Planning and Learning," 16-mm sound film, Stanford Research Institute, Menlo Park, California.

3. R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Some New Directions in Robot Problem Solving," Machine Intelligence 7 (to be published).

4. J. F. Rulifson, J. A. Derksen, and R. J. Waldinger, "QA4: A Procedural Calculus for Intuitive Reasoning," Artificial Intelligence Center Technical Note 73, Stanford Research Institute, Menlo Park, California (November 1972).

5. E. Sacerdoti, "Definition and Use of Abstraction Spaces by a Problem Solver," internal memorandum, Stanford Research Institute, Menlo Park, California (May 17, 1972).

6. C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," Artificial Intelligence, Vol. 1, No. 3, pp. 205-226 (1970).

7. D. Nitzan, "Scene Analysis Using Range Data," Artificial Intelligence Center Technical Note 69, Stanford Research Institute, Menlo Park, California (August 1972).

8. W. M. Wichman, "Use of Optical Feedback in the Computer Control of an Arm," AI Project Memo No. AI-56, Stanford University, Stanford, California (August 1967).

9. G. Falk, "Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene," AI Project Memo No. AIM-132, Stanford University, Stanford, California (August 1970).

10. B. M. Wilber, "A Shakey Primer," Artificial Intelligence Center Memo, Stanford Research Institute, Menlo Park, California (December 1972).